Heuristic construction of codeword stabilized codes

Alex Rigby [*], J. C. Olivier, and Peter Jarvis

College of Sciences and Engineering, University of Tasmania, Hobart, Tasmania 7005, Australia

The family of codeword stabilized codes encompasses the stabilizer codes as well as many of the best known nonadditive codes. However, constructing optimal n-qubit codeword stabilized codes is made difficult by two main factors. The first of these is the exponential growth with n of the number of graphs on which a code can be based. The second is the NP-hardness of the maximum clique search required to construct a code from a given graph. We address the second of these issues through the use of a heuristic clique finding algorithm. This approach has allowed us to find $((9, 97 \leqslant K \leqslant 100, 2))$ and $((11, 387 \leqslant K \leqslant 416, 2))$ codes, which are larger than any previously known codes. To address the exponential growth of the search space, we demonstrate that graphs that give large codes typically yield clique graphs with a large number of nodes. The number of such nodes can be determined relatively efficiently, and we demonstrate that n-node graphs yielding large clique graphs can be found using a genetic algorithm. This algorithm uses a crossover operation based on spectral bisection that we demonstrate to be superior to more standard crossover operations. Using this genetic algorithm approach, we have found $((13, 18, 4))$ and $((13, 20, 4))$ codes that are larger than any previously known code. We also consider codes for the amplitude damping channel. We demonstrate that for $n \leqslant 9$, optimal codeword stabilized codes correcting a single amplitude damping error can be found by considering standard form codes that detect one of only three of the 3^n possible equivalent error sets. By combining this error-set selection with the genetic algorithm approach, we have found $((11, 68))$ and $((11, 80))$ codes capable of correcting a single amplitude damping error and $((11, 4))$, $((12, 4))$, $((13, 8))$, and $((14, 16))$ codes capable of correcting two amplitude damping errors.

I. INTRODUCTION

Quantum codes can be used to protect quantum information against the effects of a noisy channel. An n-qubit code is a subspace $\mathcal{Q} \subseteq (\mathbb{C}^2)^{\otimes n}$ of dimension K. If $\mathcal{Q}$ can detect any arbitrary error on fewer than d qubits, but not some error on d qubits, then $\mathcal{Q}$ is said to have distance d and is called an $((n, K))$ or $((n, K, d))$ code. Equivalently, a code has distance d if it can detect the set $\mathcal{E}$ of all Pauli errors of weight less than d but cannot detect some weight-d Pauli error [1]. A well-understood family of codes is the stabilizer (additive) codes, which are codes defined using an Abelian subgroup of the n-qubit Pauli group [2]. However, codes outside of the stabilizer framework, called nonadditive codes, can potentially encode a larger subspace while still detecting the same error set [3–8]. Codeword stabilized (CWS) codes encompass both the stabilizer codes and many of the best known nonadditive codes [9,10]. In general, an $((n, K))$ CWS code is defined using an n-qubit stabilizer state, which is a stabilizer code of dimension one, and a set of K n-qubit Pauli operators called word operators [9]. A standard form CWS code $\mathcal{Q}$ is one where the stabilizer state is defined by a simple

undirected n-node graph G (that is, it is a graph state) and the word operators are defined by a binary classical code $\mathcal{C} \subseteq \mathrm{GF}(2)^n$ with $|\mathcal{C}| = K$ [9]. For $\mathcal{Q}$ to detect an error set $\mathcal{E}$, $\mathcal{C}$ must detect the classical error set $Cl_G(\mathcal{E}) \subseteq \mathrm{GF}(2)^n$ induced by the graph. An appropriate classical code of maximum size can be found by constructing a clique graph and performing a maximum clique search [10].

The error set that must be detected by an $((n, K, d))$ code $\mathcal{Q}$ is invariant under any permutation of the Pauli matrices X, Y, and Z on any subset of qubits. As a result of this symmetry, we call $((n, K, d))$ codes symmetric codes. This symmetry also means that if $\mathcal{Q}'$ is local Clifford (LC) equivalent to $\mathcal{Q}$ (that is, if $\mathcal{Q}' = U\mathcal{Q}$ for some LC operator U), then $\mathcal{Q}'$ is also an $((n, K, d))$ code. It follows from the LC equivalence of every stabilizer state to a graph state [11–13] that every CWS code is LC equivalent to one in standard form [9]. It is therefore sufficient to consider only standard form codes when attempting to construct an optimal $((n, K, d))$ CWS code. In fact, it is sufficient to consider only codes based on graph states that are not LC equivalent up to a permutation of qubit labels [10]. This corresponds to considering only graphs that are not isomorphic up to a series of local complementations [11]. For $n \leqslant 12$, this set of inequivalent graphs, denoted $\mathcal{L}_n$, has been enumerated [14–16] and in theory can be exhaustively searched to construct an optimal code. Such a search of $\mathcal{L}_{10}$ has previously yielded the well-known $((10, 24, 3))$ code [5]. For distance-two codes, searching $\mathcal{L}_n$ quickly becomes prohibitive with increasing n due to the rapidly growing clique graphs and the NP-hardness of finding a maximum clique [17]. To address this, we employ the heuristic

*alex.rigby@utas.edu.au

Phased Local Search (PLS) clique finding algorithm [18]. Using this approach, we have found $((9, 97 \leqslant K \leqslant 100, 2))$ and $((11, 387 \leqslant K \leqslant 416, 2))$ codes that are larger than the best known nonadditive codes presented in Refs. [6] and [7], respectively.

The apparent exponential growth of $|\mathcal{L}_n|$ with increasing n means that even if $\mathcal{L}_n$ were enumerated for $n \geqslant 13$, an exhaustive search would be prohibitive. As such, other search strategies are required for constructing codes. To aid this search, we demonstrate a relationship between the code size and the order (number of nodes) of the clique graph yielded by a given graph. In particular, we show that the clique graph orders exhibit clustering and that the graphs yielding the best codes tend to belong to the highest clique graph order cluster. This reduces the search to finding graphs that yield large clique graphs, and we show that such graphs can be generated by using a genetic algorithm to search the set of all distinct n-node graphs. This genetic algorithm uses a crossover operation based on spectral bisection, which we show to be significantly more effective than standard single-point, two-point, and uniform crossover operations. Using this genetic algorithm approach, we have found $((13, 18, 4))$ and $((13, 20, 4))$ codes. These codes are larger than an optimal $((13, 16, 4))$ stabilizer code, and to the best of our knowledge they are the first $d \geqslant 4$ codes to achieve this (we note that there is a family of $d = 8$ nonadditive codes that are larger than the best known, but not necessarily optimal, stabilizer codes [8]).

For asymmetric codes, the error set $\mathcal{E}$ that they detect is no longer invariant under Pauli matrix permutation. This means that if $\mathcal{Q}$ detects $\mathcal{E}$, then there is no guarantee that an LC-equivalent code $\mathcal{Q}' = U\mathcal{Q}$ also detects $\mathcal{E}$. However, if $\mathcal{Q}$ detects the LC-equivalent error set $U^\dagger \mathcal{E} U$, then $\mathcal{Q}'$ will detect $\mathcal{E}$. As a result, when attempting to construct an optimal $((n, K))$ CWS code detecting $\mathcal{E}$, it is sufficient to consider standard form codes based on elements of $\mathcal{L}_n$ that detect one of the up to 6^n possible LC-equivalent error sets [19] (the 6^n value stems from there being six possible permutations of the Pauli matrices on each of the n qubits). Such an asymmetric error set arises when constructing codes that correct amplitude damping errors. In this case, a partial symmetry reduces the number of LC-equivalent error sets to 3^n; however, this is still large enough to make an exhaustive search prohibitive for $n \geqslant 10$. Again, we therefore require different search strategies for constructing codes. We demonstrate that for $n \leqslant 9$, optimal CWS codes correcting a single amplitude damping error can be found by considering only codes based on nonisomorphic graphs that detect one of three LC-equivalent error sets. By combining this error-set selection with the genetic algorithm approach, we have found $((11, 68))$ and $((11, 80))$ codes capable of correcting a single amplitude damping error. These are larger than the best known stabilizer codes detecting the same error set [2]. We have also found $((11, 4))$, $((12, 4))$, $((13, 8))$, and $((14, 16))$ stabilizer codes capable of correcting two amplitude damping errors.

The paper is organized as follows. Section II gives an introduction to undirected graphs, genetic algorithms, classical codes, and quantum codes. Section III details our search strategies for symmetric codes and presents the best codes we have found. This is then extended to asymmetric codes
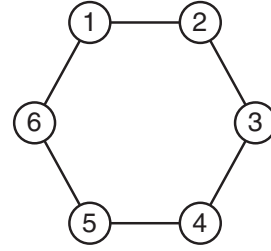

FIG. 1. A drawing of a cycle graph where the circles correspond to nodes and the lines to edges.

for the amplitude damping channel in Sec. IV. The paper is concluded in Sec. V.

II. BACKGROUND

A. Undirected graphs

A simple undirected graph $G = (N, E)$ of order n consists of a set of nodes $N = \{v_1, v_2, \ldots, v_n\}$ and a set of edges $E \subseteq \{\{v_i, v_j\} : v_i, v_j \in N, v_i \neq v_j\}$. An edge $e = \{v_i, v_j\} \in E$ is an unordered pair that connects the nodes $v_i, v_j \in N$, which are called the endpoints of e. A graph is typically drawn with the nodes depicted as circles that are joined by lines representing the edges. An example of such a drawing is given in Fig. 1. G can be represented by the symmetric $n \times n$ adjacency matrix Γ where

$$\Gamma_{ij} = \begin{cases} 1 & \text{if } \{v_i, v_j\} \in E, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

The neighborhood $\mathcal{N}(v_i) = \{v_j : \{v_i, v_j\} \in E\}$ of some node $v_i \in N$ is the set of nodes to which it is connected. The degree $\deg(v_i) = |\mathcal{N}(v_i)|$ of v_i is the number of nodes to which it is connected. The $n \times n$ degree matrix D has elements

$$D_{ij} = \begin{cases} \deg(v_i) & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

A subgraph $G_S(N_S, E_S)$ of $G = (N, E)$ is a graph with nodes $N_S \subseteq N$ and edges $E_S \subseteq E$. The subgraph induced by a subset of nodes $N_I \subseteq N$ is the graph $G_I = G[N_I] = (N_I, E_I)$ where $E_I = \{\{v_i, v_j\} \in E : v_i, v_j \in N_I\}$ contains all the edges in E that have both endpoints in N_I. A walk is a sequence whose elements alternate between connected nodes and the edges that connect them. For example, $1, \{1, 2\}, 2, \{2, 3\}, 3, \{3, 4\}, 4, \{3, 4\}, 3$ is a walk in the graph shown in Fig. 1. The length of a walk is the number of edges it contains. A path is a walk containing no repeated nodes or edges with the exception that the first and last node can be the same, in which case the path is called a cycle. A graph such as the one shown in Fig. 1 where all nodes belong to a single cycle is called a cycle graph. A graph is connected if there is a path between any two of its nodes. A connected component of G is a maximal connected subgraph $G_S(N_S, E_S)$ [maximal in that there is no other connected subgraph $G_T(N_T, E_T)$ where $N_S \subset N_T$].

Two graphs $G_1 = (N_1, E_1)$ and $G_2 = (N_2, E_2)$ are isomorphic if they differ only up to a relabeling of nodes. Formally, they are isomorphic if there exists an isomorphism from G_1 to G_2, which is a bijection $f : N_1 \rightarrow N_2$ such that $\{v_i, v_j\} \in E_1$ if

TABLE I. The number of distinct, nonisomorphic, and non-LC-isomorphic graphs with $n \leqslant 12$ nodes.

n	$|\mathcal{D}_n|$	$|\mathcal{G}_n|$	$|\mathcal{L}_n|$
1	2^0	1	1
2	2^1	2	2
3	2^3	4	3
4	2^6	11	6
5	2^{10}	34	11
6	2^{15}	156	26
7	2^{21}	1 044	59
8	2^{28}	12 346	182
9	2^{36}	274 668	675
10	2^{45}	12 005 168	3 990
11	2^{55}	1 018 997 864	45 144
12	2^{66}	165 091 172 592	1 323 363

and only if $\{f(v_i), f(v_j)\} \in E_2$. An isomorphism $f : N \to N$ from a graph $G = (N, E)$ to itself is called an automorphism. The set of all all automorphisms of G forms a group $\text{Aut}(G)$ under composition. There are a number of packages, such as NAUTY [20,21], available for determining the automorphism group of a given graph. We denote the set of all distinct n-node graphs with nodes $N = \{1, 2, \dots, n\}$ as $\mathcal{D}_n$, the size of which grows exponentially with $|\mathcal{D}_n| = 2^{\binom{n}{2}}$. $\mathcal{D}_n$ can be partitioned up to isomorphism to give the set $\mathcal{G}_n$; $|\mathcal{G}_n|$ also grows exponentially with n [22], as shown in Table I for $n \leqslant 12$. The size of some $g \in \mathcal{G}_n$ with representative $G \in \mathcal{D}_n$ is $n!/|\text{Aut}(G)|$ [22].

The complement $\bar{G} = (N, \bar{E})$ of a graph $G = (N, E)$ has an edge $\{v_i, v_j\} \in \bar{E}$ if and only if $\{v_i, v_j\} \notin E$. A local complementation (LC) at node v_i replaces the induced subgraph $G[\mathcal{N}(v_i)]$ with its complement (while we use LC for both local Clifford and local complementation, its meaning should be clear from the context in which it is used). If two graphs $G_1, G_2 \in \mathcal{D}_n$ differ by a series of local complementations, then we say they are LC equivalent. If a series of local complementations applied to G_1 yield a graph G_2' that is isomorphic to G_2, then we say that G_1 and G_2 are LC isomorphic. Partitioning $\mathcal{D}_n$ up to LC isomorphism gives the set $\mathcal{L}_n$, which has been enumerated for $n \leqslant 12$ [14–16] and also seems to grow exponentially with n as shown in Table I. Any two graphs that are isomorphic are necessarily LC isomorphic, and, therefore, any element $l \in \mathcal{L}_n$ is the union $l = \cup_i g_i$ of elements $g_i \in \mathcal{G}_n$. These g_i can be determined from any representative of l using Algorithm 5.1 of Ref. [14]. If a subset $A \subseteq \mathcal{D}_n$ contains graphs that are representatives of m different elements of $\mathcal{G}_n$ ($\mathcal{L}_n$), then we say m of the graphs in A are nonisomorphic (non-LC isomorphic).

A graph $G = (N, E)$ is complete if every node is connected to every other node; that is, if $E = \{\{v_i, v_j\} : v_i, v_j \in N, v_i \neq v_j\}$. If an induced subgraph $G[\tilde{N}]$ for some $\tilde{N} \subseteq N$ is complete, then $\tilde{N}$ is called a clique. A clique of maximum size in G is called a maximum clique. Finding a maximum clique in a graph is is NP-hard [17]; however, there are a number of heuristic algorithms that can find large, if not maximum, cliques. One such algorithm is the Phased Local Search (PLS) [18], which performs well compared to other heuristic algorithms in terms of both speed and clique finding

ability [23]. The PLS algorithm constructs a clique by initially selecting a node at random. It then iteratively selects nodes to add to the current clique (potentially replacing an existing node in the clique) until a maximum number of selections is reached. To ensure good performance on graphs with varying structures, PLS cycles through multiple different selection methods. The search is repeated for a prescribed number of attempts, after which the largest clique found is returned.

A bipartition of $G = (N, E)$ divides the nodes into two disjoint subsets N_1 and N_2. A bipartition is called a bisection if $|N_1| = |N_2|$ for even $|N|$ or if $||N_1| - |N_2|| = 1$ for odd $|N|$. An optimal bisection is one that minimizes the number of edges connecting nodes in N_1 to those in N_2. Finding such an optimal bisection is NP-hard [24]; however, approximate heuristic approaches are available. One such approach is spectral bisection [25–28], which is based on the graph's Laplacian matrix $L = D - \Gamma$. L is positive semidefinite and as such has real, non-negative eigenvalues. The eigenvector $\boldsymbol{u} = (u_1, \dots, u_n)$ corresponding to the second smallest eigenvalue is called the Fiedler vector [29]. The Fiedler vector can be used to bisect N, with the indices of the $\lfloor n/2 \rfloor$ smallest components of $\boldsymbol{u}$ dictating the nodes in N_1 and N_2 simply being $N_2 = N \backslash N_1$.

B. Genetic algorithms

Suppose we wish to determine which element in a set A is optimal in some sense. This can be expressed as finding the $a \in A$ that maximizes a fitness function $f : A \to \mathbb{R}$. The brute-force approach to this problem is to determine the fitness of every element $a \in A$. This is called an exhaustive search and quickly becomes impractical if the search space A is large and/or evaluating the fitness of elements is computationally intensive. In such cases, heuristic search algorithms can be used to find good, but potentially not optimal, elements of A. The simplest such approach is a random search, where fitness is calculated only for the elements in a randomly selected subset $B \subset A$. Another heuristic search strategy is the genetic algorithm, which is inspired by natural evolution [30,31]. There are many genetic algorithm variants; a simple implementation is as follows. Initially, the child population, which is an N element subset of A, is randomly generated. This is followed by a calculation of each child's fitness (a child being an element of the child population). The genetic algorithm then iterates for some predetermined number of maximum generations. In each generation, the previous generation's child population becomes the current generation's parent population (whose elements are called parents). A new child population is then formed by selecting two parents at a time and producing two children from them. The parents are selected according to their fitness, with high fitness parents having a higher chance of selection. With probability p_c, the two children will be produced via crossover, which combines attributes of the two parents; otherwise, they will simply be duplicates of their parents. Each child is then subjected to mutation (random alteration) with probability p_m before being added to the child population. Once the child population again contains N children, their fitnesses are calculated and a new generation begins.

Tournament selection is a simple and commonly used method of selecting parents based on their fitness. First, a

subset of the parent population is chosen at random, and then the fittest parent within this subset is selected. The size of the subset chosen is called the tournament size; it controls the selection pressure of the genetic algorithm, which is a measure of how dependent selection is on having high fitness. If the tournament size is large, then there is high selection pressure, meaning that the highest fitness parents tend to be selected. This exploitative approach gives faster convergence; however, the search is more likely to become stuck at a suboptimal local maximum [32]. Conversely, a small tournament size will lead to greater exploration of the search space at the cost of potentially slow convergence. A common modification to the genetic algorithm is the inclusion of elitist selection, which involves adding some number of the fittest parents to the child population at the start of each generation. This preserves the best elements; however, the increased selection pressure can again increase the probability of convergence to a suboptimal local maximum.

The crossover and mutation operations used depend on how elements of A are represented. A standard representation involves encoding elements as bit strings of fixed length b. A common and simple mutation operation in this case involves flipping any given bit in a child bit string with some probability (this probability is often taken to be $1/b$ [31]). Standard crossover methods include single-point, two-point, and uniform crossover. In single-point crossover, an index $1 \leqslant i \leqslant b$ is chosen, and the values beyond this point are exchanged between the two parent bit strings to form two child bit strings. In two-point crossover, two such indices are selected and all values between them are exchanged. In uniform crossover, each individual bit is exchanged between the two parents with some probability p_e.

In some cases, representations other than bit strings are more natural. For example, it may be possible to represent elements as graphs. Crossover becomes more complicated with such a representation. A potential method is presented in Ref. [33] and is as follows. First, each of the parent graphs P_1 and P_2 are each split into two subgraphs, called fragments, to produce disconnected parents P_{1D} and P_{2D}. To split a parent graph, first an edge $\{v_i, v_j\}$ is chosen at random. In an iterative process, the shortest path between v_i and v_j is determined, and a randomly selected edge in this path is removed (in the first iteration, this will simply be the edge $\{v_i, v_j\}$). This continues until no path exists between v_i and v_j. The connected component containing v_i is the fragment F_1, and the subgraph induced by the remaining nodes is the fragment F_2. In the next step, disconnected children C_{1D} and C_{2D} are formed by exchanging a fragment, say F_1, between each of the parent graphs. The two fragments in each disconnected child are then combined to produce children C_1 and C_2. This combination process involves iteratively selecting a node from each fragment and joining them with an edge. The probability of a node being selected is proportional to the difference in its current degree to its degree in its initial parent graph. This process of adding edges is repeated until all of the nodes in one of the fragments, say F_1, have the same degree as they did in their initial parent graph. If a node v_l in F_2 has degree lower than its initial degree by some amount δ_l, then in a process repeated δ_l times, it will be connected to a randomly selected node in F_1 with 50% probability. As outlined in Ref. [34],

the splitting process presented here has some undesirable attributes. Firstly, it tends to produce two fragments with a vastly different number of nodes. Secondly, it often removes a large number of edges from within the larger fragment; these are edges that did not have to be removed to split the parent graph.

C. Classical codes

A classical channel is a map $\Phi : \mathcal{A}_x \to \mathcal{A}_y$, where $\mathcal{A}_x$ is the set of possible inputs and $\mathcal{A}_y$ is the set of possible outputs. We are concerned with channels where the input and outputs are binary [that is, channels for which $\mathcal{A}_x = \mathcal{A}_y = \mathrm{GF}(2)$]. In this case, the action of the channel can be expressed as

$$\Phi(x) = x + e = y, \tag{3}$$

where $x \in \mathrm{GF}(2)$ is the channel input, $y \in \mathrm{GF}(2)$ is the channel output, and $e \in \mathrm{GF}(2)$ is an error (or noise) symbol. A code can be used to protect against the noise introduced by the channel. A length-n binary code is a subset $\mathcal{C} \subseteq \mathrm{GF}(2)^n$ whose elements are called codewords. Codewords are transmitted as n sequential uses of Φ or, equivalently, as a single use of the combined channel Φ^n, which is comprised of n copies of Φ. The action of Φ^n on some input $\boldsymbol{x} \in \mathcal{C}$ is

$$\Phi^n(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{e} = \boldsymbol{y}, \tag{4}$$

where $\boldsymbol{y} \in \mathrm{GF}(2)^n$ is the channel output and $\boldsymbol{e} \in \mathrm{GF}(2)^n$ is an error "vector." The weight of an error is the number of nonzero components from which it comprised.

We say that a code $\mathcal{C}$ can detect a set of errors $\mathcal{E} \subseteq \mathrm{GF}(2)^n$ if

$$\boldsymbol{x}_i + \boldsymbol{e} \neq \boldsymbol{x}_j \tag{5}$$

for all $\boldsymbol{e} \in \mathcal{E}$ and $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{C}$, where $\boldsymbol{x}_i \neq \boldsymbol{x}_j$. That is, the errors in $\mathcal{E}$ can be detected if they do not map one codeword to another. Furthermore, we say that $\mathcal{C}$ can correct $\mathcal{E}$ if

$$\boldsymbol{x}_i + \boldsymbol{e}_k \neq \boldsymbol{x}_j + \boldsymbol{e}_l \tag{6}$$

for all $\boldsymbol{e}_k, \boldsymbol{e}_l \in \mathcal{E}$ and $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{C}$, where $\boldsymbol{x}_i \neq \boldsymbol{x}_j$. This condition simply ensures that two codewords cannot be mapped to the same $\boldsymbol{y} \in \mathrm{GF}(2)^n$, in which case the transmitted codeword cannot be inferred with certainty. $\mathcal{C}$ is said to have distance d if it can detect any error of weight less than d but is unable to detect some weight-d error. Note that $\mathcal{C}$ can correct $\mathcal{E}$ if and only if it can detect $\mathcal{E} + \mathcal{E} = \{\boldsymbol{e}_k + \boldsymbol{e}_l : \boldsymbol{e}_k, \boldsymbol{e}_l \in \mathcal{E}\}$, meaning that a distance-d code can correct any error of weight $t = \lfloor (d - 1)/2 \rfloor$ or less. A length-n code $\mathcal{C}$ of size $|\mathcal{C}| = K$ and distance d is called an (n, K) or (n, K, d) code. If $\mathcal{C}$ forms a vector space, then it is called linear and has $K = 2^k$. A linear code encodes the state of k bits and is called an $[n, k]$ or $[n, k, d]$ code.

Finding a code $\mathcal{C}$ of maximum size that detects an error set $\mathcal{E}$ can be expressed as a clique finding problem. This is achieved by constructing a graph $G_{\mathcal{E}} = (N_{\mathcal{E}}, E_{\mathcal{E}})$ whose nodes are potential codewords; that is, $N_{\mathcal{E}} = \mathrm{GF}(2)^n$. Two nodes $\boldsymbol{x}_i, \boldsymbol{x}_j \in N_{\mathcal{E}}$ are connected by an edge $\{\boldsymbol{x}_i, \boldsymbol{x}_j\} \in E_{\mathcal{E}}$ if $\boldsymbol{x}_i + \boldsymbol{x}_j \notin \mathcal{E}$ (that is, if there is not an error mapping one to the other). Any clique $\mathcal{C}$ in $G_{\mathcal{E}}$ is a code detecting $\mathcal{E}$, and a code of maximum possible size is a maximum clique in $G_{\mathcal{E}}$. Note that if a code $\mathcal{C}$ detects $\mathcal{E}$, then so does $\mathcal{C}' = \boldsymbol{x} + \mathcal{C}$ for any $\boldsymbol{x} \in \mathcal{C}$.

As $\mathbf{0} \in \mathcal{C}'$ and $|\mathcal{C}| = |\mathcal{C}'|$, this means there is always an optimal code (that is, a maximum size code detecting $\mathcal{E}$) that contains the all-zero codeword. The clique search can be restricted to such codes by taking $N_{\mathcal{E}} = \mathbf{0} \cup [\mathrm{GF}(2)^n \backslash \mathcal{E}]$.

D. Quantum codes

The action of a quantum channel Φ on a quantum state described by the density operator ρ is

$$\Phi(\rho) = \sum_k A_k \rho A_k^\dagger, \qquad (7)$$

where the A_k, called Kraus operators, satisfy $\sum_k A_k^\dagger A_k = I$ (the identity operator) [35]. The channel can be interpreted as mapping $\rho \mapsto A_k \rho A_k^\dagger$ (up to normalization) with probability $\mathrm{tr}(A_k \rho A_k^\dagger)$ [36]. If $\rho = |\phi\rangle\langle\phi|$ (that is, if the input state is pure), then this becomes the mapping $|\phi\rangle \mapsto A_k|\phi\rangle$ (up to normalization) with corresponding probability $\langle\phi|A_k^\dagger A_k|\phi\rangle$. In this paper, we are interested in qubit systems; that is, systems where states $|\phi\rangle$ belong to a two-dimensional Hilbert space $\mathcal{H} \cong \mathbb{C}^2$. Similar to the classical case, the noise introduced by a quantum channel can be protected against by employing a code. A quantum (qubit) code of length n is a subspace $\mathcal{Q} \subseteq (\mathbb{C}^2)^{\otimes n}$. Codewords $|\phi\rangle \in \mathcal{Q}$ are transmitted across the combined n-qubit channel $\Phi^{\otimes n}$.

Suppose a code $\mathcal{Q}$ has an orthonormal basis $\mathcal{B} = \{|\phi_1\rangle, \dots, |\phi_K\rangle\}$, and take $\mathcal{E} = \{E_1, \dots, E_r\}$ to be the basis for some complex vector space of linear n-qubit operators (called error operators). We say that $\mathcal{Q}$ can detect any error in the span of $\mathcal{E}$ if

$$\langle\phi_i|E|\phi_j\rangle = C_E \delta_{ij} \qquad (8)$$

for all $E \in \mathcal{E}$ and $|\phi_i\rangle, |\phi_j\rangle \in \mathcal{B}$, where C_E is a scalar that depends only on E [2]. Furthermore, we say that $\mathcal{Q}$ can correct any error in the span of $\mathcal{E}$ if

$$\langle\phi_i|E_k^\dagger E_l|\phi_j\rangle = C_{kl} \delta_{ij} \qquad (9)$$

for all $E_k, E_l \in \mathcal{E}$ and $|\phi_i\rangle, |\phi_j\rangle \in \mathcal{B}$, where C is an $r \times r$ Hermitian matrix [1]. The weight of an error E is the number of qubits on which it acts. $\mathcal{Q}$ has distance d if it can detect any error of weight less than d but not some weight-d error. Similar to the classical case, a code can correct $\mathcal{E}$ if and only if it can detect $\mathcal{E}^\dagger \mathcal{E} = \{E_k^\dagger E_l : E_k, E_l \in \mathcal{E}\}$, meaning that a distance-d quantum code can also correct any error of weight $t = \lfloor (d-1)/2 \rfloor$ or less. A length-n code of dimension K and distance d is called an $((n, K))$ or $((n, K, d))$ code (the double brackets differentiate from the classical case). A code $\mathcal{Q}$ correcting $\mathcal{E}$ is called nondegenerate if the spaces $E_k \mathcal{Q}$ and $E_l \mathcal{Q}$ are linearly independent (that is, their intersection is trivial) for any $E_k, E_l \in \mathcal{E}$, where $E_k \neq E_l$. If all such spaces are orthogonal, then $\mathcal{Q}$ is called pure.

The Pauli matrices in the computational $\{|0\rangle, |1\rangle\}$ basis are

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \quad (10)$$

X can be viewed as a bit-flip operator as $X|0\rangle = |1\rangle$ and $X|1\rangle = |0\rangle$. Z can be viewed as a phase flip as $Z|0\rangle = |0\rangle$ and $Z|1\rangle = -|1\rangle$. $Y = iXZ$ can be viewed as a combined bit and phase flip. The Pauli matrices are Hermitian, unitary,

and anticommute with each other. Furthermore, they form the group

$$\mathcal{P}_1 = \{\pm I, \pm iI, \pm X, \pm iX, \pm Y, \pm iY, \pm Z, \pm iZ\} = \langle X, Y, Z \rangle \qquad (11)$$

called the Pauli group. The n-qubit Pauli group $\mathcal{P}_n$ consists of all n-fold tensor product combinations of elements of $\mathcal{P}_1$. For example, $\mathcal{P}_8$ contains the element $I \otimes I \otimes X \otimes I \otimes Y \otimes Z \otimes I \otimes I$, which we can write more compactly as $X_3 Y_5 Z_6$. The weight $w(g)$ of some $g \in \mathcal{P}_n$ is the number of elements in the tensor product that are not equal to the identity up to phase. The commutation relations of the Pauli matrices mean that elements of $\mathcal{P}_n$ must either commute or anticommute, with two elements anticommuting if their nonidentity components differ in an odd number of places. The Pauli matrices along with the identity form a basis for the complex vector space of all 2×2 matrices. It therefore follows that

$$\mathcal{E}^r = \{E = \sigma_1 \otimes \cdots \otimes \sigma_n : \sigma_i \in \{I, X, Y, Z\} \text{ and } w(E) \leqslant r\} \qquad (12)$$

is a basis for all n-qubit errors of weight less than or equal to r. An equivalent definition is $\mathcal{E}^r = \{E_1 \dots E_r : E_i \in \mathcal{E}^1\}$; that is, $\mathcal{E}_r$ is the set of all r-fold products of elements of $\mathcal{E}_1$, which can be written as $\mathcal{E}^1 = \{I, X_i, Y_i, Z_i\}$ where $1 \leqslant i \leqslant n$. It is sometimes convenient to express some $E \in \mathcal{P}_n$ up to phase as $E \propto X^{u_1} Z^{v_1} \otimes \cdots \otimes X^{u_n} Z^{v_n} = X^{\boldsymbol{u}} Z^{\boldsymbol{v}}$ where $\boldsymbol{u} = (u_1, \dots, u_n)$, $\boldsymbol{v} = (v_1, \dots, v_n) \in \mathrm{GF}(2)^n$.

Two n-qubit codes $\mathcal{Q}$ and $\mathcal{Q}'$ are local unitary (LU) equivalent if $\mathcal{Q}' = U\mathcal{Q}$ for some $U \in U(2)^{\otimes n}$. These codes will have the same dimension as if $\mathcal{B} = \{|\phi_1\rangle, \dots, |\phi_K\rangle\}$ is an orthonormal basis for $\mathcal{Q}$, then $\mathcal{B}' = U\mathcal{B} = \{U|\phi_1\rangle, \dots, U|\phi_K\rangle\}$ is an orthonormal basis for $\mathcal{Q}'$. It follows from Eq. (8) that $\mathcal{Q}'$ detects the error set $\mathcal{E}$ if and only if $\mathcal{Q}$ detects the LU-equivalent error set $\mathcal{E}' = U^\dagger \mathcal{E} U$. Furthermore, $\mathcal{E}$ is a basis for all errors of weight less than d if and only if $\mathcal{E}'$ is also such a basis. Therefore, $\mathcal{Q}$ and $\mathcal{Q}'$ have the same distance; that is, they are both $((n, K, d))$ codes. If two codes differ by a LU operator and/or permutation of qubit labels, which also has no effect on the size or distance of the code, then they are called equivalent codes. The normalizer of $\mathcal{P}_1$ in $U(2)$ is the single-qubit Clifford group $\mathcal{C}_1 = \{U \in U(2) : U^\dagger \mathcal{P}_1 U = \mathcal{P}_1\}$. The n-qubit local Clifford group $\mathcal{C}_1^n$ is comprised of all possible n-fold tensor products of elements from $\mathcal{C}_1$. Two codes are local Clifford (LC) equivalent if they are LU equivalent for some $U \in \mathcal{C}_1^n \subset U(2)^{\otimes n}$.

Stabilizer codes (also called additive codes) are defined by an Abelian subgroup $\mathcal{S} < \mathcal{P}_n$, called the stabilizer, that does not contain $-I$ [2]. The code $\mathcal{Q}$ is the space of states that are fixed by every element $s_i \in \mathcal{S}$; that is,

$$\mathcal{Q} = \{|\phi\rangle \in (\mathbb{C}^2)^{\otimes n} : s_i|\phi\rangle = |\phi\rangle \; \forall \, s_i \in \mathcal{S}\}. \qquad (13)$$

The requirement that $-I \notin \mathcal{S}$ both means that no $s \in \mathcal{S}$ can have a phase factor of $\pm i$, and that if $s \in \mathcal{S}$, then $-s \notin \mathcal{S}$. If $\mathcal{S}$ is generated by $M = \{M_1, \dots, M_m\} \subset \mathcal{P}_n$, then it is sufficient (and obviously necessary) for $\mathcal{Q}$ to be stabilized by every M_i. Assuming that the set of generators is minimal, it can be shown that $\dim(\mathcal{Q}) = 2^{n-m} = 2^k$ [36]; that is, $\mathcal{Q}$ encodes the state of a k-qubit system. An n-qubit stabilizer code with dimension $K = 2^k$ and distance d is called an $[[n, k]]$ or $[[n, k, d]]$ code.

An n-qubit stabilizer state $|\mathcal{S}\rangle$ is an $[[n, 0, d]]$ code defined by a stabilizer $\mathcal{S}$ with n generators. The distance of a stabilizer state is defined to be equal to the weight of the lowest nonzero weight element in $\mathcal{S}$. A graph state $|G\rangle$ is a stabilizer state defined by a graph $G \in \mathcal{D}_n$. Each node i corresponds to a qubit and is also associated with a stabilizer generator

$$M_i = X_i Z_{\mathcal{N}(i)} = X_i \prod_{j \in \mathcal{N}(i)} Z_j. \tag{14}$$

Each graph state $|G\rangle$ defines a basis $\mathcal{B} = \{Z^{\boldsymbol{w}}|G\rangle : \boldsymbol{w} \in \mathrm{GF}(2)^n\}$ for $(\mathbb{C}^2)^{\otimes n}$ [37]. An error $E = X_i$ maps the graph state $|G\rangle$ to

$$X_i|G\rangle = X_i(X_i Z_{\mathcal{N}(i)})|G\rangle = Z_{\mathcal{N}(i)}|G\rangle = Z^{\boldsymbol{r}_i}|G\rangle, \tag{15}$$

where $\boldsymbol{r}_i$ is the ith row of the adjacency matrix for G. That is, an X error applied at node i is equivalent to Z errors being applied at its neighbors; this is called the $X - Z$ rule [38]. It can be shown that every stabilizer state is LC equivalent to a graph state [11–13]. Two graph states $|G_1\rangle$ and $|G_2\rangle$ are the same up to a relabeling of qubits if and only if their corresponding graphs G_1 and G_2 are isomorphic. Furthermore, $|G_1\rangle$ and $|G_2\rangle$ are LC equivalent if and only if G_1 and G_2 are LC equivalent [11]. Therefore, $|G_1\rangle$ and $|G_2\rangle$ are equivalent (as quantum codes) if G_1 and G_2 are LC isomorphic (the converse does not necessarily hold as two states can be LU equivalent without being LC equivalent [39]).

E. CWS codes

The family of codeword stabilized (CWS) codes contains all stabilizer codes as well as many of the best known nonadditive codes [9,10]. An $((n, K))$ CWS code $\mathcal{Q}$ is defined using an n-qubit stabilizer state $|\mathcal{S}\rangle$ and a set of K word operators $\mathcal{W} = \{W_1, \ldots, W_K\} \subset \mathcal{P}_n$. In particular, $\mathcal{Q}$ is the span of the basis codewords $|W_i\rangle = W_i|\mathcal{S}\rangle$. Note that for the $|W_i\rangle$ to actually form a basis, no two word operators can differ only by a stabilizer element; that is, it cannot be the case that $W_i W_j \in \bar{\mathcal{S}} = \cup_{\alpha \in \{\pm 1, \pm i\}} \alpha \mathcal{S}$ for any $W_i \neq W_j$. For a CWS code, the criterion for detecting an error set $\mathcal{E}$ becomes

$$\langle W_i|E|W_j\rangle = \langle \mathcal{S}|W_i^\dagger E W_j|\mathcal{S}\rangle = C_E \delta_{ij} \tag{16}$$

for all $E \in \mathcal{E}$ and $W_i, W_j \in \mathcal{W}$. If $\mathcal{E}$ contains only Pauli errors $E \in \mathcal{P}_n$, then

$$\langle \mathcal{S}|W_i^\dagger E W_j|\mathcal{S}\rangle = \begin{cases} 0 & \text{if } W_i^\dagger E W_j \notin \bar{\mathcal{S}}, \\ \alpha & \text{if } W_i^\dagger E W_j \in \alpha \mathcal{S}, \end{cases} \tag{17}$$

where $\alpha \in \{\pm 1, \pm i\}$. Therefore, the $i \neq j$ case of Eq. (16) holds for some $E \in \mathcal{E}$ if and only if

$$W_i^\dagger E W_j \notin \bar{\mathcal{S}} \tag{18}$$

for all $W_i, W_j \in \mathcal{W}$, where $W_i \neq W_j$. Furthermore, the $i = j$ case holds for some $E \in \mathcal{E}$ if and only if either

$$W_i^\dagger E W_i \notin \bar{\mathcal{S}} \tag{19}$$

or

$$W_i^\dagger E W_i \in \alpha \mathcal{S} \tag{20}$$

for all $W_i \in \mathcal{W}$ and some particular $\alpha \in \{\pm 1, \pm i\}$.

It follows from the LC equivalence of every stabilizer state to a graph state that every CWS code is LC equivalent to one

based on a graph state $|G\rangle$ with word operators of the form $W_i = Z^{\boldsymbol{x}_i}$ [9]. Such a code is called a standard form CWS code, and its basis codewords are simply elements of the graph basis defined by G. The set $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K\} \subseteq \mathrm{GF}(2)^n$ forms a classical binary code $\mathcal{C}$, and without loss of generality we can take $\boldsymbol{x}_1 = \boldsymbol{0}$ [9]. It can be shown that if $\mathcal{C}$ is linear, then the CWS code is additive [9], whereas if $\mathcal{C}$ is not linear, then the code may be additive or nonadditive [10] (although if $K \neq 2^k$, then the CWS code must obviously be nonadditive). The effect of an error $E \propto X^{\boldsymbol{u}} Z^{\boldsymbol{v}}$ on one of the basis codewords $|W_i\rangle = Z^{\boldsymbol{x}_i}|G\rangle$ follows from the $X - Z$ rule with

$$\begin{aligned} E|W_i\rangle &\propto X^{\boldsymbol{u}} Z^{\boldsymbol{v}} Z^{\boldsymbol{x}_i}|G\rangle \\ &\propto Z^{\boldsymbol{v}} Z^{\boldsymbol{x}_i} X^{\boldsymbol{u}}|G\rangle \\ &= Z^{\boldsymbol{v}} Z^{\boldsymbol{x}_i} Z^{\boldsymbol{u}\Gamma}|G\rangle \\ &= Z^{\boldsymbol{v}} Z^{\boldsymbol{u}\Gamma} Z^{\boldsymbol{x}_i}|G\rangle \\ &= Z^{Cl_G(E)}|W_i\rangle, \end{aligned} \tag{21}$$

where Γ is the adjacency matrix for G and

$$Cl_G(E \propto X^{\boldsymbol{u}} Z^{\boldsymbol{v}}) = \boldsymbol{v} + \boldsymbol{u}\Gamma. \tag{22}$$

Therefore, the effect of $E \propto X^{\boldsymbol{u}} Z^{\boldsymbol{v}}$ is equivalent to that of $E' = Z^{Cl_G(E)}$, where $Cl_G(E) \in \mathrm{GF}(2)^n$ is a classical error induced by the graph. It follows from this equivalence that $\langle W_i|E|W_j\rangle \propto \langle W_i|Z^{Cl_G(E)}|W_j\rangle$, which means that Eq. (18) is satisfied when

$$Z^{\boldsymbol{x}_i} Z^{Cl_G(E)} Z^{\boldsymbol{x}_j} \notin \bar{\mathcal{S}}. \tag{23}$$

For a graph state, the only stabilizer element with no X component is the identity $I = Z^{\boldsymbol{0}}$. Equation (23) therefore reduces to $\boldsymbol{x}_i + Cl_G(E) \neq \boldsymbol{x}_j$, which is simply the classical error detection criterion of Eq. (5). This means that an error E can be detected only if $\mathcal{C}$ detects the classical error $Cl_G(E)$. Following the same reasoning, Eq. (19) becomes $\boldsymbol{x}_i + Cl_G(E) \neq \boldsymbol{x}_i$, which reduces to $Cl_G(E) \neq \boldsymbol{0}$. Equation (20) becomes

$$Z^{\boldsymbol{x}_i} E Z^{\boldsymbol{x}_i} \in \alpha \mathcal{S}, \tag{24}$$

which reduces to $E \in \alpha \mathcal{S}$ for $\boldsymbol{x}_1 = \boldsymbol{0}$. If there is some $W_i = Z^{\boldsymbol{x}_i}$ that anticommutes with E, then Eq. (24) becomes $E \in -\alpha \mathcal{S}$. This would mean that both $\alpha^{-1} E \in \mathcal{S}$ and $-\alpha^{-1} E \in \mathcal{S}$, from which it follows that $-I \in \mathcal{S}$. This cannot be the case as $\mathcal{S}$ is a stabilizer. Therefore, to satisfy Eq. (20), it must be the case that $[Z^{\boldsymbol{x}_i}, E] = 0$ for all $\boldsymbol{x}_i \in \mathcal{C}$. For $E \propto X^{\boldsymbol{u}} Z^{\boldsymbol{v}}$, this condition is equivalent to requiring $\boldsymbol{x}_i \cdot \boldsymbol{u} = 0$ for all $\boldsymbol{x}_i \in \mathcal{C}$, were $\boldsymbol{a} \cdot \boldsymbol{b} = \sum_i a_i b_i$ is the standard Euclidean inner product. In summary, a standard form code detects $E \propto X^{\boldsymbol{u}} Z^{\boldsymbol{v}} \in \mathcal{E}$ if

$$\boldsymbol{x}_i + Cl_G(E) \neq \boldsymbol{x}_j \tag{25}$$

for all $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{C}$, where $\boldsymbol{x}_i \neq \boldsymbol{x}_j$, and either

$$Cl_G(E) \neq \boldsymbol{0} \tag{26}$$

or

$$\boldsymbol{x}_i \cdot \boldsymbol{u} = 0 \tag{27}$$

for all $\boldsymbol{x}_i \in \mathcal{C}$.

Designing a CWS code $\mathcal{Q}$ for a given graph G and error set $\mathcal{E}$ consists of finding a classical code $\mathcal{C}$ that satisfies

Eqs. (25)–(27) for every $E \in \mathcal{E}$. It is convenient to express this as a clique finding problem as outlined in Ref. [10]. First, the set of classical errors

$$Cl_G(\mathcal{E}) = \{Cl_G(E) : E \in \mathcal{E}\} \qquad (28)$$

induced by the graph is determined. Also required is the set

$$D_G(\mathcal{E}) = \{\mathbf{x} \in \mathrm{GF}(2)^n : Cl_G(E) = \mathbf{0} \text{ and}$$
$$\mathbf{x} \cdot \mathbf{u} \neq 0 \text{ for some } E \propto X^{\mathbf{u}} Z^{\mathbf{v}} \in \mathcal{E}\}. \qquad (29)$$

These are elements of $\mathrm{GF}(2)^n$ that cannot be included in the code as they violate Eqs. (26) and (27). An algorithm for efficiently determining $Cl_G(\mathcal{E})$ and $D_G(\mathcal{E})$ is given in Ref. [10]. A classical code $\mathcal{C}$ satisfying Eqs. (25)–(27) is a clique in the graph $G_\mathcal{E} = (N_\mathcal{E}, E_\mathcal{E})$ with

$$N_\mathcal{E} = \mathbf{0} \cup \{\mathrm{GF}(2)^n \backslash [Cl_G(\mathcal{E}) \cup D_G(\mathcal{E})]\} \qquad (30)$$

and $E_\mathcal{E}$ defined by the classical error set $Cl_G(\mathcal{E})$ as outlined in Sec. II C. That is, two nodes $\mathbf{x}_i, \mathbf{x}_j \in N_\mathcal{E}$ are connected by an edge $\{\mathbf{x}_i, \mathbf{x}_j\} \in E_\mathcal{E}$ if $\mathbf{x}_i + \mathbf{x}_j \notin Cl_G(\mathcal{E})$. If $D_G(\mathcal{E}) = \emptyset$, then for all $E \not\propto I \in \mathcal{E}$ it must be the case that $Cl_G(E) \neq \mathbf{0}$, and hence $C_E = 0$ in Eq. (16). Therefore, for $E = E_k^\dagger E_l \in \mathcal{E}$ where $E_k E_l \in \mathcal{P}_n$ and $E_k \not\propto E_l$, it follows that $\langle W_i | E_k^\dagger E_l | W_j \rangle = 0$. That is, $\mathcal{Q}$ is pure if $D_G(\mathcal{E}) = \emptyset$ [10,40].

F. Code bounds

A simple, but relatively loose, upper bound on the dimension K of an n-qubit code of distance d is given by the quantum singleton bound [1]

$$K \leqslant 2^{n-2(d-1)}. \qquad (31)$$

A tighter limit on code size is given by the linear programming bound [41]. An $((n, K, d))$ code can exist only if there are homogeneous polynomials $A(x, y)$, $B(x, y)$, and $S(x, y)$ such that

$$A(1, 0) = 1, \qquad (32)$$

$$B(x, y) = KA\left(\frac{x + 3y}{2}, \frac{x - y}{2}\right), \qquad (33)$$

$$S(x, y) = KA\left(\frac{x + 3y}{2}, \frac{y - x}{2}\right), \qquad (34)$$

$$B(1, y) - A(1, y) = O(y^d), \qquad (35)$$

$$A(x, y) \geqslant 0, \qquad (36)$$

$$B(x, y) - A(x, y) \geqslant 0, \qquad (37)$$

$$S(x, y) \geqslant 0. \qquad (38)$$

Here, $C(x, y) \geqslant 0$ means that the coefficients of the polynomial C are non-negative, and $O(y^d)$ is a polynomial in y with no terms of degree less than d. A pure $((n, K, d))$ code can exist only if Eqs. (32)–(38) can be satisfied along with

$$A(1, y) = 1 + O(y^d). \qquad (39)$$

The linear programming bound is monotonic [42], meaning that if the constraints can be satisfied for some K, then they can be satisfied for all lower code dimensions too. This monotonicity holds even if K is allowed to be a real number (rather than just an integer). Following Ref. [43], we define the real

TABLE II. Bounds on the maximum k of an $[[n, k, d]]$ stabilizer code for $1 \leqslant n \leqslant 15$ and $2 \leqslant d \leqslant 5$.

$n\backslash d$	2	3	4	5
1				
2	0			
3	0			
4	2			
5	2	1		
6	4	1^C	0	
7	4	1	$-^A$	
8	6	3	0	
9	6	3	0	
10	8	4	2	
11	8	5	2	1
12	10	6	4	1^C
13	10	7	4^B	1
14	12	8	6	2−3
15	12	9	6^A	3^A

number $K(n, d)$ as the largest $K > 1$ for which Eqs. (32)–(38) can be satisfied. The purity conjecture of Ref. [41] states that if the linear programming constraints hold for $K = K(n, d)$, then $A(1, y) = 1 + O(y^d)$. The content of this conjecture is simply that the linear programming bound for pure codes is the same as for potentially impure codes. This conjecture has been verified to hold for $n \leqslant 100$ [43].

For stabilizer codes, bounds on maximum k are given in Table II for $1 \leqslant n \leqslant 15$ and $2 \leqslant d \leqslant 5$. All lower bounds are given by the best known stabilizer codes (these codes can be found at Ref. [44]). The unmarked upper bounds are given by the linear programming bound for $K = 2^k$ (determined using YALMIP [45]). If the lower and upper bounds coincide, then a single value is given; otherwise, they are separated by a dash. In the cases marked "A," the [[7,0,4]], [[15,7,4]], and [[15,4,5]] codes that do not violate the linear programming bound are excluded by arguments given in Sec. 7 of Ref. [41]. In the case marked "B," the [[13,5,4]] code that does not violate the linear programming bound is excluded by the argument of Ref. [46]. The entries marked "C" indicate cases where a code meeting the bound must be impure (also outlined in Sec. 7 of Ref. [41]). An extended version of Table II for $n \leqslant 256$ is available at Ref. [44].

Table III gives the bounds on maximum K for a potentially nonadditive $((n, K, d))$ code where $1 \leqslant n \leqslant 15$ and $2 \leqslant d \leqslant 5$. All upper bounds are from the linear programming bound. The lower bounds marked "A" are from the family of nonadditive $((2\alpha + 1, 3 \times 2^{2\alpha-3}, 2))$ codes of Ref. [6]. Those marked "B" are from the family of $((4\alpha + 2\beta + 3, M_{\alpha\beta}, 2))$ codes of Ref. [7] where $\beta \in \{0, 1\}$ and

$$M_{\alpha\beta} = \sum_{i=0}^{\alpha} \binom{4\alpha + 2\beta + 3}{2i + \beta}. \qquad (40)$$

The lower bounds marked "C" and "D" correspond to the $((9,12,3))$ and $((10,24,3))$ codes of Refs. [4] and [5], respectively. All other lower bounds are given by the best known stabilizer codes.

TABLE III. Bounds on the maximum size K of an $((n, K, d))$ code for $1 \leqslant n \leqslant 15$ and $2 \leqslant d \leqslant 5$.

$n \backslash d$	2	3	4	5
1				
2	1			
3	1			
4	4			
5	^{A}6	2		
6	16	2	1	
7	^{A}24–26	2–3	0–1	
8	64	8–9	1	
9	^{A}96–112	^{C}12–13	1	
10	256	^{D}24	4–5	
11	^{B}386–460	32–53	4–7	2
12	1 024	64–89	16–20	2
13	^{B}1 586–1 877	128–204	16–40	2–3
14	4 096	256–324	64–102	4–10
15	^{B}6 476–7 606	512–580	64–150	8–18

III. SYMMETRIC CODES

An $((n, K, d))$ code must detect the set $\mathcal{E}^{d-1}$ as defined in Eq. (12). Note that $\mathcal{E}^1$, and hence $\mathcal{E}^{d-1}$ more generally, is invariant under any permutation of the Pauli matrices X, Y, and Z on any subset of qubits. As a result of this symmetry, we call $((n, K, d))$ codes symmetric codes. Furthermore, as outlined in Sec. II D, this symmetry means that if some code $\mathcal{Q}$ detects $\mathcal{E}^{d-1}$, then so does any equivalent code $\mathcal{Q}'$. It is therefore sufficient to consider only standard form codes when attempting to construct an optimal symmetric CWS code. Furthermore, we need only consider standard form codes based on representatives from different elements of $\mathcal{L}_n$. However, as outlined in Sec. II A, the size of $\mathcal{L}_n$ appears to grow exponentially, and it has only been enumerated for $n \leqslant 12$. Furthermore, constructing an optimal classical code for a given graph by finding a maximum clique is NP-hard as mentioned in Sec. II A. In this section we explore methods of code construction that address these two obstacles.

A. Distance-two codes

First we consider distance-two codes of even length. As outlined in Tables II and III, there are even length stabilizer codes with $k = n - 2$ that saturate the singleton bound for $n \leqslant 14$. In fact, there are stabilizer codes that saturate the bound for all even n [6]. Despite this, there is still some insight to be gained from constructing CWS codes with these parameters. For $n \leqslant 10$, it is feasible to exhaustively search $\mathcal{L}_n$ (that is, to construct a code based on a representative of each element of $\mathcal{L}_n$). Using the code size distribution over $\mathcal{L}_n$, it is possible to determine the distributions over $\mathcal{G}_n$ and $\mathcal{D}_n$ by counting the number of nonisomorphic and distinct graphs, respectively, in each element of $\mathcal{L}_n$ (see Sec. II A). As an example, the code size distributions for $n = 6$ are shown in Fig. 2. It can be seen that over 50%, 75%, and 80% of elements of $\mathcal{L}_6$, $\mathcal{G}_6$, and $\mathcal{D}_6$, respectively, yield optimal $K = 16$ codes. The fraction of elements of $\mathcal{L}_n$, $\mathcal{G}_n$, and $\mathcal{D}_n$ that yield optimal codes for even $2 \leqslant n \leqslant 10$ is shown in Table IV. For $2 \leqslant n \leqslant 6$, the clique graphs generated are small enough for maximum cliques to

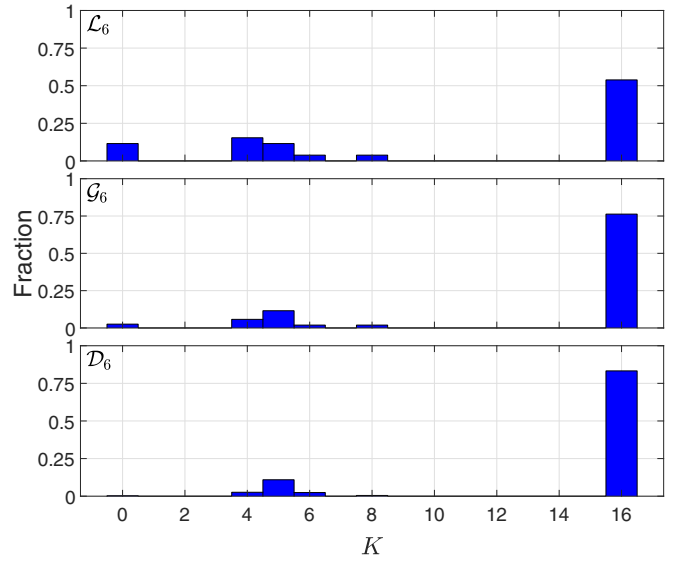

FIG. 2. Code size distributions for non-LC-isomorphic, nonisomorphic, and distinct graphs in the case of $n = 6$ and $d = 2$.

be found using the exact algorithm of Ref. [47]. For $n \geqslant 8$, we have resorted to using the approximate PLS algorithm due to the larger clique graphs. We have allowed the PLS algorithm 100 attempts, each of which used a maximum of 1000 selections (these are the default PLS parameters that we have employed). As a result of having used an approximate clique finding algorithm, the values given in the $n = 8$ and 10 rows of Table IV are a lower bounds. It can be seen that in each case, the fraction of elements in $\mathcal{D}_n$ yielding an optimal code is greater than that of $\mathcal{G}_n$, which in turn is greater than that of $\mathcal{L}_n$. Furthermore, increasing n increases the fraction of optimal codes in all cases. In particular, by $n = 10$ over 98% of distinct graphs yield a code with an optimal $K = 256$. This trend suggests that for larger n, we are highly likely to find an optimal code even if we use a randomly selected graph. This goes some way to explaining the results of Ref. [48], where cycle graphs were shown to give optimal codes for even $n \leqslant 12$.

The case of odd n is somewhat more interesting. Here, as shown in Ref. [6], the linear programming bound reduces to

$$K \leqslant 2^{n-2} \left(1 - \frac{1}{n-1} \right). \tag{41}$$

Stabilizer codes cannot saturate this bound and are restricted to $k \leqslant n - 3$. Again, we can construct codes based on an

TABLE IV. The fraction of elements of $\mathcal{L}_n$, $\mathcal{G}_n$, and $\mathcal{D}_n$ that yield optimal $K = 2^{n-2}$ codes for even $n \leqslant 10$ and $d = 2$. The values given for $n = 8$ and 10 are lower bounds.

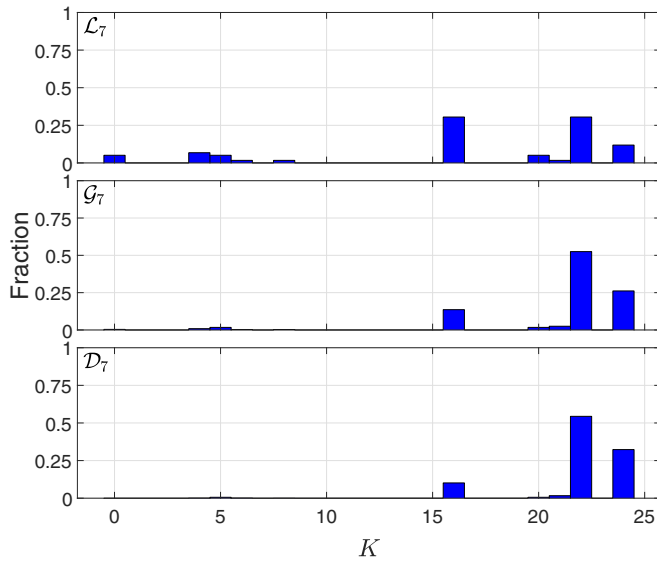| n	$\mathcal{L}_n$	$\mathcal{G}_n$	$\mathcal{D}_n$
2	0.500	0.500	0.500
4	0.500	0.636	0.641
6	0.539	0.763	0.833
8	0.643	0.909	0.938
10	0.815	0.977	0.981

FIG. 3. Code size distributions for non-LC-isomorphic, nonisomorphic, and distinct graphs in the case of $n = 7$ and $d = 2$.

TABLE V. Number of elements N_K of $\mathcal{L}_{11}$ that gave codes of given size K with $d = 2$.

K	387	388	389	390	391	392	398	400	402	404	406	408	416
N_K	51	11	1	1	2	54	2	207	1	74	1	6	2

exhaustive search of $\mathcal{L}_n$ for $n \leqslant 11$. For $n = 3$, a single element of $\mathcal{L}_3$ yields an optimal $K = 1$ code. Similarly, a single element of $\mathcal{L}_5$ yields a code with $K = 6$, which matches the size of the optimal code given in Refs. [3,6]. For $n = 7$, there is more of a spread in the code sizes as shown in Fig. 3. It can be seen that a large number of graphs yield codes with $K = 16$ or 22, which match the size of an optimal stabilizer code and the code of Ref. [7], respectively. Furthermore, there are seven elements of $\mathcal{L}_7$ that yield codes with $K = 24$, which match the size of the code of Ref. [6]. No graphs yield codes with $K = 25$ or 26, despite such codes not being excluded by the linear programming bound.

For $n = 9$, an exhaustive search of $\mathcal{L}_9$ is still feasible; however, we have done so using the PLS clique finder, and as such there may exist larger CWS codes than the ones reported here. Similar to the $n = 7$ case, the majority of graphs gave
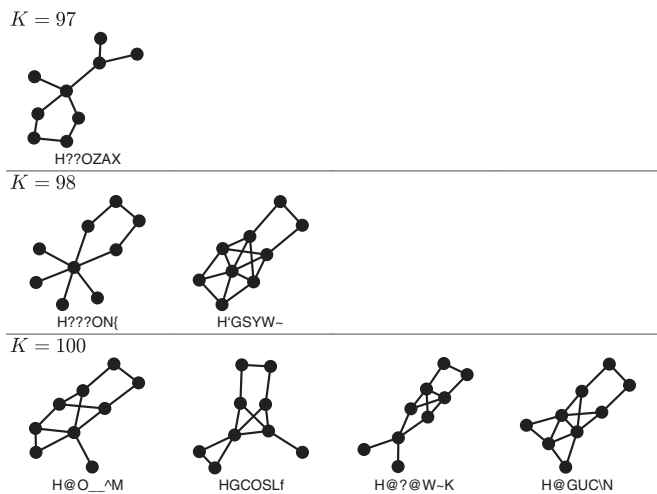
codes with $K = 64$, 93, or 96, which match the size of an optimal stabilizer code, the code of Ref. [7], and the code of Ref. [6], respectively. However, we have also found seven elements of $\mathcal{L}_9$ that yield codes with $K \geqslant 97$. To increase the likelihood that we have found maximum size codes for these seven graphs, we have repeated the clique search for each of them using 10 000 attempts. This has resulted in one $K = 97$ code, two $K = 98$ codes, and four $K = 100$ codes. Representatives of the elements of $\mathcal{L}_n$ that yielded these codes are shown in Fig. 4. Note that we do not label the nodes as isomorphic graphs yield equivalent codes. Given below each of the drawings is the graph in graph6 format (see Ref. [21] for details). A classical code for each of these graphs is given in the Supplemental Material [49] (this is the case for all codes presented in this paper). While these $K \geqslant 97$ codes are larger than any previously known codes, they do not saturate the linear programming bound of $K = 112$.

For $n = 11$, we have performed an exhaustive search of $\mathcal{L}_{11}$ with an increased 10 000 PLS selections to account for the larger cliques. Here, we have mostly obtained codes with $K = 256$, 384, or 386, which match the size of an optimal stabilizer code, the code of Ref. [6], and the code of Ref. [7], respectively. We have also found 413 elements of $\mathcal{L}_{11}$ that yield codes with $K \geqslant 387$. As for the $n = 9$ case, we have repeated the clique search for these graphs using 10 000 attempts. The resulting code size distribution is given in
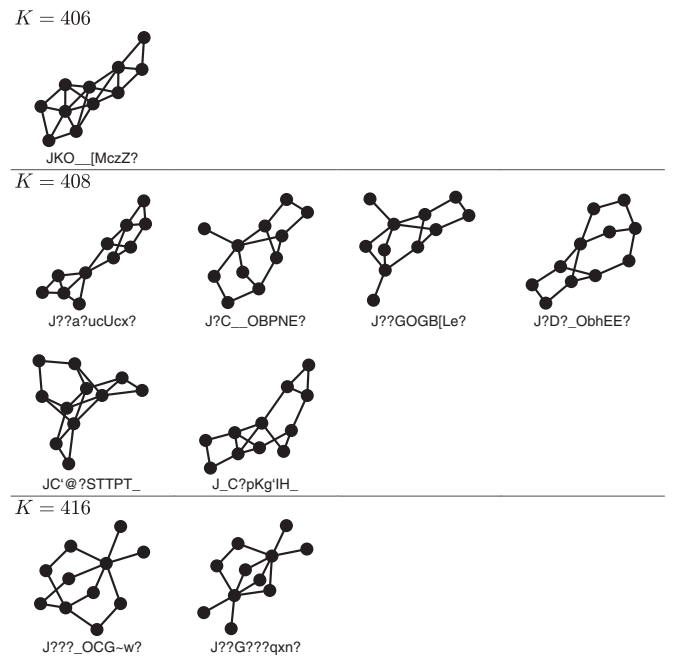

FIG. 4. Non-LC-isomorphic graphs that yield $((9, 97 \leqslant K \leqslant 100, 2))$ codes.

FIG. 5. Non-LC-isomorphic graphs that yield $((11, 406 \leqslant K \leqslant 416, 2))$ codes.

FIG. 6. Code size versus clique graph order for codes with $4 \leqslant n \leqslant 11$ and $d = 2$.
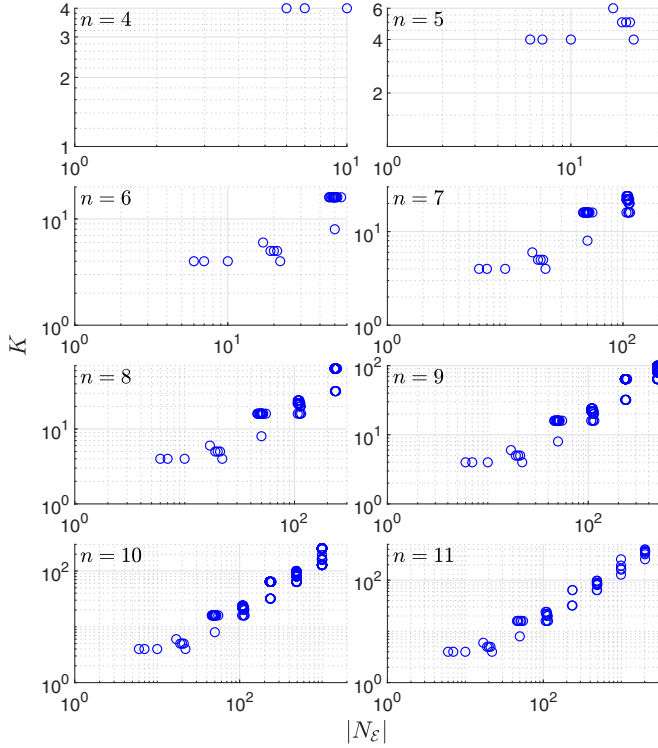

FIG. 7. Code size versus clique graph order for codes with $8 \leqslant n \leqslant 11$ and $d = 3$.

Table V. Representatives of elements of $\mathcal{L}_{11}$ that yield codes with $K \geqslant 406$ are shown in Fig. 5 (the remaining graphs are included in the Supplemental Material [49]). Again, while these are the largest codes known, they do not saturate the linear programming bound of $K = 460$. As $\mathcal{L}_n$ has not been enumerated for $n \geqslant 13$, we cannot continue this exhaustive search procedure for higher n. Any (nonexhaustive) search of $\mathcal{G}_n$ or $\mathcal{D}_n$ is also impractical for $n \geqslant 13$ due to the large clique graphs produced, which both makes the clique search slow and reduces the likelihood that the clique found is of maximum size.

Figure 6 shows the relationship between code size and clique graph order $|N_{\mathcal{E}}|$ for $4 \leqslant n \leqslant 11$. It can be seen that the data are clustered by clique graph order; furthermore, in each case, the graphs yielding the largest codes belong to the highest $|N_{\mathcal{E}}|$ cluster. This clustering behavior can be explained by considering Eq. (30), which gives

$$|N_{\mathcal{E}}| = 2^n + 1 - |Cl_G(\mathcal{E})| - |D_G(\mathcal{E})|. \quad (42)$$

It follows from Eq. (29) that $\mathrm{GF}(2)^n \backslash D_G(\mathcal{E})$ is the annihilator of $\mathcal{E}' = \{E \in \mathcal{E} : Cl_G(E) = 0\}$ and is therefore a subspace of $\mathrm{GF}(2)^n$. If $\dim[\mathrm{GF}(2)^n \backslash D_G(\mathcal{E})] = r \leqslant n$, then $|D_G(\mathcal{E})| = 2^n - 2^r$, which gives $|N_{\mathcal{E}}| = 2^r - |Cl_G(\mathcal{E})|$. The clusters therefore correspond to different values of r. The codes in the highest $|N_{\mathcal{E}}|$ cluster are pure as they have $D_G(\mathcal{E}) = \emptyset$. That this cluster contains codes of maximum size is not entirely surprising in light of the purity conjecture outlined in Sec. II F.
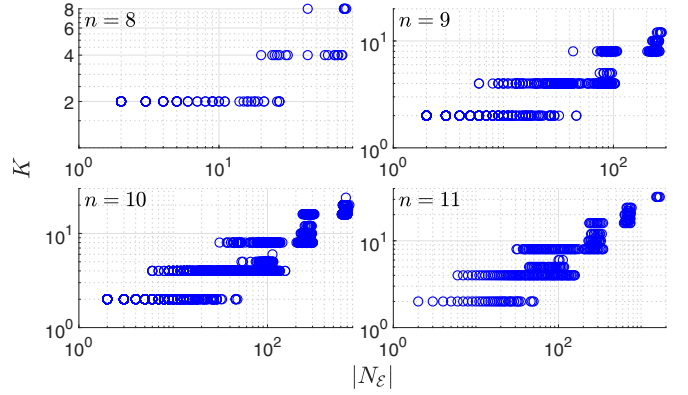
B. Distance-three codes

Distance-three codes are of practical interest as they allow for the correction of an arbitrary single-qubit error. For $n \leqslant 11$, we can exhaustively search $\mathcal{L}_n$ in the same way as we have for the distance-two codes of the previous section. There are one and two elements of $\mathcal{L}_5$ and $\mathcal{L}_6$, respectively, that give optimal $K = 2$ codes (note that all $K = 2$ CWS codes are additive [10]). Similarly, there are 18 elements of $\mathcal{L}_7$ that yield $K = 2$ codes. As has been previously shown in Ref. [10], although the linear programming bound does not exclude them, there are no ((7, 3, 3)) CWS codes. There are six elements of $\mathcal{L}_8$ that give $K = 8$ codes. No elements yield a $K = 9$ code, despite such a code not being excluded by the linear programming bound. There are eight elements of $\mathcal{L}_9$ that yield $K = 12$ codes, which match the size of the code presented in Ref. [4]. Again, no elements yield a $K = 13$ code, despite such a code not being excluded by the linear programming bound. An exhaustive search of $\mathcal{L}_{10}$ has previously been performed in Ref. [5], where it was shown that a single element yields an optimal $K = 24$ code. We have exhaustively searched $\mathcal{L}_{11}$ using the PLS clique finder. This has yielded 13 709 $K = 32$ codes, which match the size of an optimal stabilizer code. No larger codes were found, which is somewhat surprising given that the linear programming bound is $K = 53$.

Figure 7 shows the relationship between code size and clique graph order for distance-three codes with $8 \leqslant n \leqslant 11$. It can be seen that there is greater spread within the clusters compared to the distance-two case of Fig. 6. According to Eq. (42), this can be attributed to an increased variance in the size of $Cl_G(\mathcal{E})$. Despite this increased variation, the graphs yielding the best codes still belong to the highest $|N_{\mathcal{E}}|$ cluster in all four cases. Importantly, the best codes are not necessarily given by the graphs with the highest clique graph order within this cluster. For example, in the $n = 10$ case, the highest clique graph cluster contains graphs with $613 \leqslant |N_{\mathcal{E}}| \leqslant 739$, while the graph yielding the $K = 24$ code only has $|N_{\mathcal{E}}| = 679$.

For $n = 12$, the size of $\mathcal{L}_{12}$ makes an exhaustive search somewhat prohibitive. We can reduce the search space somewhat by considering the distribution of clique graph orders as shown in Fig. 8. Note that by using Eq. (42), $|N_{\mathcal{E}}|$ can be
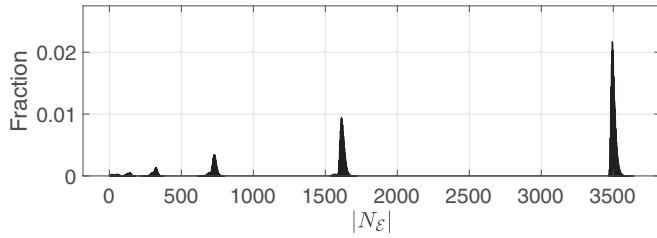
FIG. 8. Clique graph order distribution over $\mathcal{L}_{12}$ for codes with $d = 3$.

FIG. 9. Clique graph order distribution over $\mathcal{D}_{13}$ for codes with with $d = 4$.

computed without actually constructing the clique graph. Our previous observations regarding the relationship between code size and clique graph order suggest that graphs yielding the best codes are highly likely to be found in the $|N_\mathcal{E}| > 3000$ cluster. We have randomly selected 50 000 of the 663 039 elements of $\mathcal{L}_{12}$ in this cluster and constructed a code for each using the PLS clique finder. This has yielded 6325 codes with $K = 64$, which match the size of an optimal stabilizer code. No larger codes were found, despite the linear programming bound not excluding codes with up to $K = 89$. We have not pursued searches for $n \geqslant 13$ codes as while the clique graphs produced are smaller than in the $d = 2$ case, they are still large enough for maximum clique searches to be unreliable.

C. Distance-four codes

For $d = 4$, we are able to perform exhaustive searches of $\mathcal{L}_n$ for $n \leqslant 12$. There are one, five, and eight elements of $\mathcal{L}_6$, $\mathcal{L}_8$, and $\mathcal{L}_9$, respectively, that yield optimal $K = 1$ codes. As expected, no elements of $\mathcal{L}_7$ give a nontrivial code (note that a $K = 1$ CWS code is a stabilizer state and hence pure; such $[[n, 0, d]]$ codes have previously been classified in [15]). There are 10 and 3060 elements of $\mathcal{L}_{10}$ and $\mathcal{L}_{11}$, respectively, that give $K = 4$ codes, which match the size of an optimal stabilizer code. No elements yield larger codes, despite the linear programming bound not excluding codes with up to $K = 5$ and 7, respectively. Unlike the $d = 3$ case, an exhaustive search of $\mathcal{L}_{12}$ is feasible for $d = 4$ due to the smaller clique graphs. However, the clique graphs are still large enough that we have resorted to using the PLS clique finding algorithm. This search has yielded 1482 codes with $K = 16$, which match the size of an optimal stabilizer code. No larger codes were found, despite the linear programming bound not excluding codes with up to $K = 20$. The smaller clique graph sizes in the $d = 4$ case also make searching for codes with $n = 13$ and 14 feasible. For $n = 13$, we have randomly selected 100 000 graphs from $\mathcal{D}_{13}$ to estimate the clique graph size distribution as shown in Fig. 9. 41 458 of these graphs belong to the $|N_\mathcal{E}| > 2000$ cluster. Of these, one yielded a $K = 18$ code, which is larger than an optimal $K = 16$ stabilizer code.

To find more $n = 13$ codes with $K > 16$, we want a more reliable way of generating graphs that yield a large clique graph. That is, we wish to search $\mathcal{D}_n$ for graphs yielding a large clique graph in a way that is more efficient than a random search. We have found a genetic algorithm to be effective in this respect. There are a number of ways we could implement mutation and crossover in this algorithm. For mutation, we
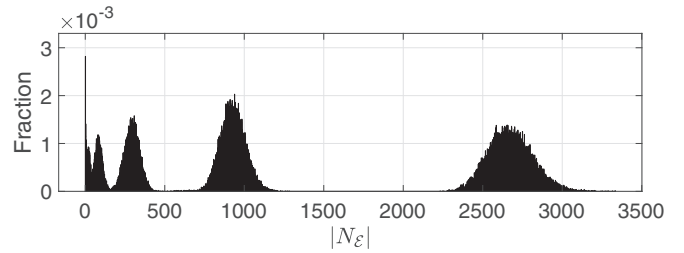
first select two nodes in the child graph at random. If these two nodes are not connected by an edge, then one is added; otherwise, if they are connected by an edge, then it is removed. If we represent the parent graphs as bit strings, then we can use standard single-point, two-point, or uniform crossover. One way to achieve this is to convert the upper triangular component of a parent adjacency matrix to a bit string row by row. Alternatively, we can use a graph-based approach. However, the method of Ref. [33] outlined in Sec. II B is not appropriate for searching $\mathcal{D}_n$ as it is not guaranteed to produce child graphs with n nodes. Furthermore, as previously mentioned, it tends to remove an unnecessarily large number of edges when splitting the parent graphs into two fragments. To address these issues, we propose splitting the parent graphs using a spectral bisection. In particular, the nodes of a parent graph P are bisected into the sets N_1 and N_2, which define the fragments $F_1 = P[N_1]$ and $F_2 = P[N_2]$. A fragment is then exchanged between each parent to form two disconnected children that are then connected following the method of Ref. [33]. An example of this procedure on two $n = 10$ graphs is shown in Fig. 10.

We have run 100 genetic algorithm instances using each of the potential crossover methods to compare their performance. In each instance, we have used a population size of $N = 20$, 100 generations, a crossover probability of $p_c = 0.9$, a mutation probability of $p_m = 0.1$, and a tournament size of 10. We have also incorporated elitist selection, with the fittest two parent graphs (that is, the two that yield the largest clique graphs) being added to the child population at the start of each generation. The average order of the highest-order clique graph yielded in each generation is shown in Fig. 11. It can be seen that single-point, two-point, and uniform crossover (with $p_e = 0.5$) all exhibit similar performance. However, their performance is also matched by random crossover, where the two children are simply selected at random from $\mathcal{D}_n$ with no input from the parents. As such, the increase in fitness with successive generations when using these crossover methods is simply due to the selection pressure of the genetic algorithm. It can also be seen that spectral crossover gives significantly better performance than all other methods. We have also tested the effect of population size when using spectral crossover. In particular, we have tested population sizes of $N = 10$ and 40 in addition to the previously considered $N = 20$ case. We have used a tournament size of half the population size in each case and left all other parameters unchanged. It can be seen in Fig. 11 that, as expected, increasing the population size increases the average maximum fitness. With clique graph
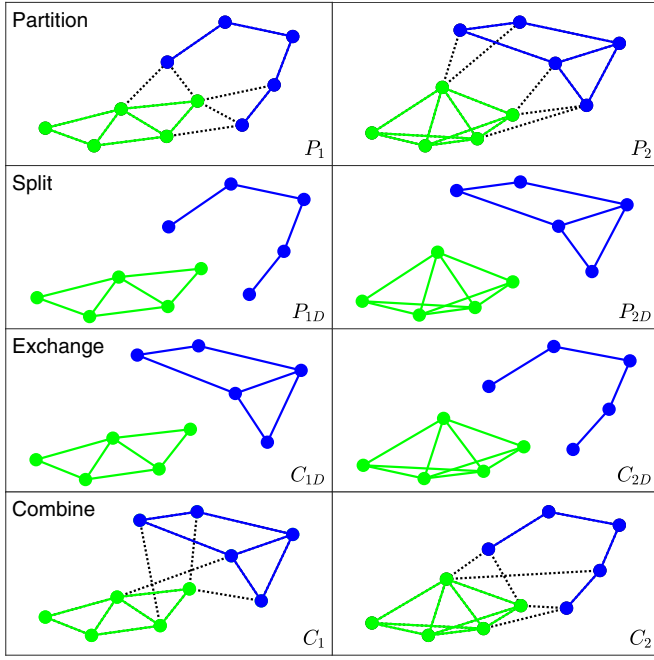
FIG. 10. Spectral crossover example for $n = 10$ graphs. Each parent graph is split into two fragments according to a spectral bisection. These fragments are then exchanged and combined to form two child graphs.

order only serving as an indicator of code size, it is not essential for the genetic algorithm to find graphs that yield the absolute largest clique graphs. In fact, as was seen in the $n = 10$, $d = 3$ case, focusing solely on such graphs may mean that we miss the best code(s). With this in mind, we have found using 50 generations and a population size of $N = 10$ to be a good compromise. Using a modest population size and number of generations is also favorable from a run time perspective as determining $|N_\mathcal{E}|$ becomes more computationally expensive with increasing code length and/or distance (both of which serve to increase the size of the error set).
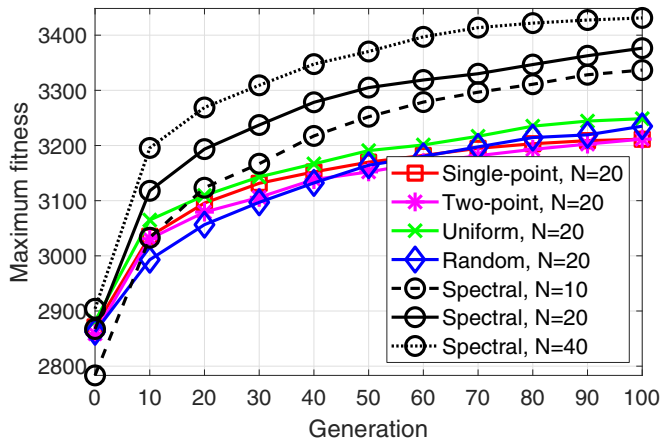

FIG. 11. Comparison of crossover methods for $n = 13$, $d = 4$ codes. The vertical axis shows the fitness (the clique graph order $|N_\mathcal{E}|$) of the highest fitness element of the child population averaged over 100 genetic algorithm instances.

The genetic algorithm we have outlined is quite exploitative. To make our search more explorative, we run a large number of genetic algorithm instances, with a code being constructed from the fittest graph found by each instance. For $n = 13$, we have run 50 000 such instances, of which 352 yielded a $K = 18$ code and a further 175 gave a $K = 20$ code. The graphs that yielded codes with $K = 18$ and 20 belong to 35 and 25 different elements of $\mathcal{L}_{13}$, respectively. A representative from each of these elements is shown in Figs. 12 and 13. Note that the graphs shown are not necessarily the exact ones found using the genetic algorithm; they are LC-equivalent graphs that can be drawn clearly using the force-directed layout method of Ref. [50]. While these $K = 18$ and 20 codes are larger than any previously known codes, they do not saturate the linear programming bound of $K = 40$. We have also run 50 000 instances of the genetic algorithm for $n = 14$. 65 of these instances have yielded $K = 64$ codes, which match the size of an optimal stabilizer code. We have not found any codes with $K > 64$, despite the linear programming bound not excluding codes with up to $K = 102$.

D. Distance-five codes

For $d = 5$, one and five elements of $\mathcal{L}_{11}$ and $\mathcal{L}_{12}$, respectively, yield optimal $K = 2$ codes. For $13 \leqslant n \leqslant 15$ we have run 50 000 genetic algorithm instances. 46 978 instances yielded a $K = 2$ code for $n = 13$, 452 instances yielded a $K = 4$ code for $n = 14$, and 14 instances yielded a $K = 8$ code for $n = 15$. No larger codes were found, despite the linear programming bound being $K = 3$, 10, and 18, respectively. Note that the existence of a $((13, 3, 5))$ CWS code has already been excluded in Ref. [10] by the same argument that excluded the $((7, 3, 3))$ code.

IV. ASYMMETRIC CODES

A channel of physical interest is the amplitude damping channel

$$\rho \rightarrow A_0 \rho A_0^\dagger + A_1 \rho A_1^\dagger, \tag{43}$$

where
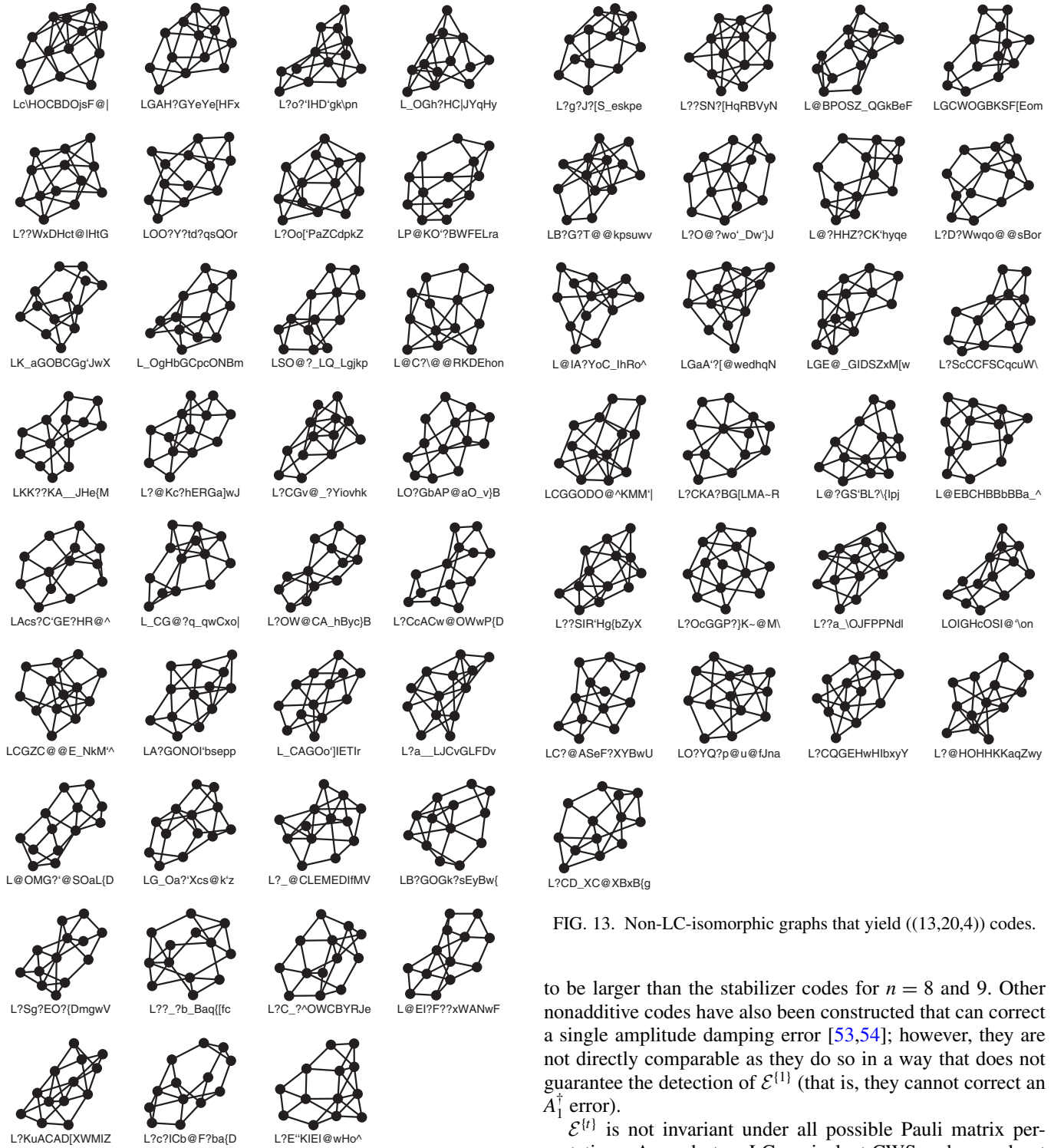
$$A_0 = \begin{pmatrix} 1 & 0 \\ 0 & \sqrt{1-\gamma} \end{pmatrix}, \quad A_1 = \begin{pmatrix} 0 & \sqrt{\gamma} \\ 0 & 0 \end{pmatrix}. \tag{44}$$

It can be shown [2,51,52] that a sufficient condition for correcting a single amplitude damping error is the ability to detect

$$\mathcal{E}^{\{1\}} = \{I, X_i, Y_i, Z_i, X_iX_j, X_iY_j, Y_iY_j\}, \tag{45}$$

where $1 \leqslant i, j \leqslant n$. This is not a necessary condition for correcting an amplitude damping error. In fact, a code detecting $\mathcal{E}^{\{1\}}$ can also correct a single $A_1^\dagger$ error [19]. A code can correct t amplitude damping errors if it can detect $\mathcal{E}^{\{t\}}$, which is comprised of all t-fold combinations of elements from $\mathcal{E}^{\{1\}}$. $\mathcal{E}^{\{t\}}$ is a subset of $\mathcal{E}^{2t}$, which is the set of errors that must be detected to guarantee the ability to correct an arbitrary weight-t error. As a result, there is potential for constructing codes correcting t amplitude damping errors that are larger than those correcting t arbitrary errors. For example,

Lc\HOCBDOjsF@| LGAH?GYeYe[HFx L?o?'IHD'gk\pn L_OGh?HC|JYqHy

L??WxDHct@IHtG LOO?Y?td?qsQOr L?Oo['PaZCdpkZ LP@KO'?BWFELra

LK_aGOBCGg'JwX L_OgHbGCpcONBm LSO@?_LQ_Lgjkp L@C?\@@RKDEhon

LKK??KA__JHe{M L?@Kc?hERGa]wJ L?CGv@_?Yiovhk LO?GbAP@aO_v}B

LAcs?C'GE?HR@^ L_CG@?q_qwCxo| L?OW@CA_hByc}B L?CcACw@OWwP{D

LCGZC@@E_NkM'^ LA?GONOl'bsepp L_CAGOo']IETlr L?a__LJCvGLFDv

L@OMG?'@SOaL{D LG_Oa?'Xcs@k'z L?_@CLEMEDlfMV LB?GOGk?sEyBw{

L?Sg?EO?{DmgwV L??_?b_Baq{[fc L?C_?^OWCBYRJe L@EI?F??xWANwF

L?KuACAD[XWMIZ L?c?ICb@F?ba{D L?E"KIEl@wHo^

FIG. 12. Non-LC-isomorphic graphs that yield ((13,18,4)) codes.

L?g?J?[S_eskpe L??SN?[HqRBVyN L@BPOSZ_QGkBeF LGCWOGBKSF[Eom

LB?G?T@@kpsuwv L?O@?wo'_Dw'}J L@?HHZ?CK'hyqe L?D?Wwqo@@sBor

L@IA?YoC_IhRo^ LGaA'?[@wedhqN LGE@_GIDSZxM[w L?ScCCFSCqcuW\

LCGGODO@^KMM'| L?CKA?BG[LMA~R L@?GS'BL?\{lpj L@EBCHBBbBBa_^

L??SIR'Hg{bZyX L?OcGGP?}K~@M\ L??a_\OJFPPNdl LOIGHcOSI@'\on

LC?@ASeF?XYBwU LO?YQ?p@u@fJna L?CQGEHwHIbxyY L?@HOHHKKaqZwy

L?CD_XC@XBxB{g

FIG. 13. Non-LC-isomorphic graphs that yield ((13,20,4)) codes.

to be larger than the stabilizer codes for $n = 8$ and 9. Other nonadditive codes have also been constructed that can correct a single amplitude damping error [53,54]; however, they are not directly comparable as they do so in a way that does not guarantee the detection of $\mathcal{E}^{\{1\}}$ (that is, they cannot correct an $A_1^\dagger$ error).

$\mathcal{E}^{\{t\}}$ is not invariant under all possible Pauli matrix permutations. As such, two LC-equivalent CWS codes need not correct the same number of amplitude damping errors. This

TABLE VI. Size of stabilizer codes presented in Ref. [2] and CWS codes presented in Ref. [19] that detect $\mathcal{E}^{\{1\}}$.

n	4	5	6	7	8	9	10	11	12	13	14	15
Stabilizer	1	2	4	8	8	16	32	64	128	256	512	1024
CWS		2	4	8	10	20						

the stabilizer codes presented in Ref. [2] detect $\mathcal{E}^{\{1\}}$ and have the parameters given in Table VI (these values are taken from Ref. [53]). In all but the $n = 8$ case, these codes are larger than the size of an optimal $d = 3$ stabilizer code as given in Table II. An exhaustive search for CWS codes detecting $\mathcal{E}^{\{1\}}$ has been performed in Ref. [19] for $5 \leqslant n \leqslant 9$. The size of these codes is also given in Table VI, where they can be seen

TABLE VII. Number of elements of $\mathcal{G}_n$ that yield optimal $((n, K))$ CWS codes for the LC-equivalent error sets $\mathcal{E}^{\{1\}}$, $\mathcal{E}_{XZ}^{\{1\}}$, and $\mathcal{E}_{YZ}^{\{1\}}$. The values given for $n = 9$ are lower bounds.

	$\mathcal{E}^{\{1\}}$	$\mathcal{E}_{XZ}^{\{1\}}$	$\mathcal{E}_{YZ}^{\{1\}}$
$((5,2))$	5	9	3
$((6,4))$	11	16	0
$((7,8))$	114	157	181
$((8,10))$	0	4	36
$((9,20))$	0	6	44

means that considering standard form codes based on different elements of $\mathcal{L}_n$ no longer constitutes an exhaustive search of all CWS codes. However, as suggested in Ref. [19], a search of $\mathcal{L}_n$ can be made exhaustive by performing it for every LC-equivalent error set of the form $U^\dagger \mathcal{E}^{\{t\}} U$. These sets are versions of $\mathcal{E}^{\{t\}}$ with X, Y, and Z errors permuted on some subset of qubits. If $\mathcal{E}^{\{t\}}$ exhibited no symmetries under such permutations, then there would be 6^n such sets. However, as $\mathcal{E}^{\{t\}}$ is invariant under the permutation $X \leftrightarrow Y$ on any subset of qubits, this number is reduced to 3^n. Unfortunately, an exhaustive search is not practical for codes with $n \geqslant 10$ as even for $n = 10$, there are $3^{10}|\mathcal{L}_{10}| = 235\,605\,510$ cases to test. In this section, we build on our code construction methods to address this increase in the size of the search space.

A. Single amplitude damping error

To construct new codes for the amplitude damping channel with $n \geqslant 10$, we first consider $n \leqslant 9$ to determine what types of codes match the bounds provided in Ref. [19]. Initially, we restrict consideration to standard form codes that detect $\mathcal{E}^{\{1\}}$. As $\mathcal{E}^{\{1\}}$ (and $\mathcal{E}^{\{t\}}$ more generally) is invariant under a permutation of qubit labels, it is sufficient to consider one representative from each element of $\mathcal{G}_n$. The first column of Table VII shows the number of elements of $\mathcal{G}_n$ for $5 \leqslant n \leqslant 9$ that yield optimal standard form CWS codes. Note that the value given for $n = 9$ is a lower bound as we have used the PLS clique finder in this case. It can be seen that while we are able to construct optimal codes for $5 \leqslant n \leqslant 7$, we are unable to do so for $n = 8$ and 9. To remedy this, we consider the LC-equivalent error sets

$$\mathcal{E}_{XZ}^{\{1\}} = \{I, X_i, Y_i, Z_i, Z_i Z_j, Z_i Y_j, Y_i Y_j\}, \quad (46)$$

$$\mathcal{E}_{YZ}^{\{1\}} = \{I, X_i, Y_i, Z_i, X_i X_j, X_i Z_j, Z_i Z_j\}. \quad (47)$$

These versions of $\mathcal{E}^{\{1\}}$ with the permutations $X \leftrightarrow Z$ and $Y \leftrightarrow Z$, respectively, on every qubit. More generally, we define $\mathcal{E}_{XZ}^{\{t\}}$ and $\mathcal{E}_{YZ}^{\{t\}}$ to be versions of $\mathcal{E}^{\{t\}}$ with the permutations $X \leftrightarrow Z$ and $Y \leftrightarrow Z$, respectively, on every qubit. Columns two and three of Table VII show that exhaustive searches of $\mathcal{G}_n$ using the error sets $\mathcal{E}_{XZ}^{\{1\}}$ and $\mathcal{E}_{YZ}^{\{1\}}$ yield optimal codes for $n = 8$ and 9.

For $n = 10$, the size of $\mathcal{G}_{10}$ combined with the sizes of the clique graphs generated makes an exhaustive search impractical. However, we can still determine the distribution of clique graph sizes over $\mathcal{G}_n$ for the three error sets $\mathcal{E}^{\{1\}}$, $\mathcal{E}_{XZ}^{\{1\}}$, and $\mathcal{E}_{YZ}^{\{1\}}$ as shown in Fig. 14. For each of the three error sets,
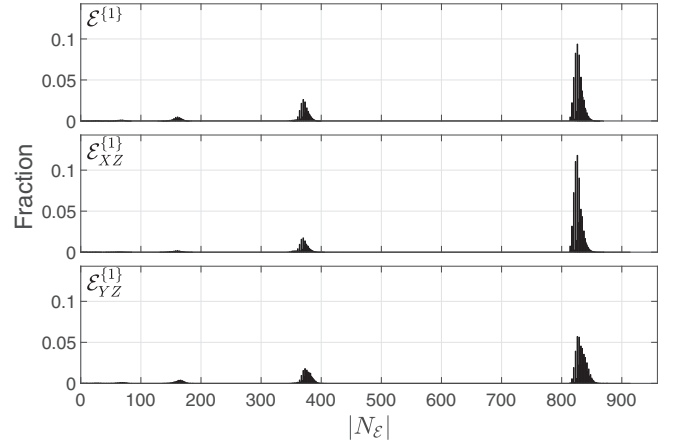

FIG. 14. Distribution of clique graph order over $\mathcal{G}_{10}$ for the error sets $\mathcal{E}^{\{1\}}$, $\mathcal{E}_{XZ}^{\{1\}}$, and $\mathcal{E}_{YZ}^{\{1\}}$.

50 000 graphs in the $|N_{\mathcal{E}}| > 600$ cluster have been selected. In each case, all 50 000 graphs yielded $K = 32$ codes, which match the size of the stabilizer code presented in Ref. [2].

For $n = 11$, an exhaustive search of $\mathcal{G}_{11}$ is impractical, even to simply determine clique graph sizes. We have therefore run 50 000 instances of our genetic algorithm for each of the three error sets $\mathcal{E}^{\{1\}}$, $\mathcal{E}_{XZ}^{\{1\}}$, and $\mathcal{E}^{\{1\}}$. For $\mathcal{E}^{\{1\}}$, this has yielded a $K = 64$ code in every case. These codes match the size of the stabilizer code presented in Ref. [2]. For $\mathcal{E}_{YZ}^{\{1\}}$, 1818 instances yielded codes with $K = 68$, which are larger than the best known stabilizer codes. 28 of these graphs are nonisomorphic and are shown in Fig. 15 (a simple circular node layout is used here as we do not have the freedom of picking an LC-isomorphic graph that can be drawn clearly using the force-directed layout method). For $\mathcal{E}_{XZ}^{\{1\}}$, only nine instances yielded codes with $K = 68$; however, there were also 71 instances that yielded codes with $K = 80$. Of these, two of the $K = 68$ graphs are nonisomorphic and two of the $K = 80$ graphs are nonisomorphic; these graphs are also shown in Fig. 15. For $n = 12$, applying the same genetic algorithm approach has yielded codes with $K = 128$, which match the size of the stabilizer code presented in Ref. [2]. In particular, of the 50 000 instances run for each error set, 21 535 gave a $K = 128$ code for $\mathcal{E}^{\{1\}}$, 34 906 gave a $K = 128$ code for $\mathcal{E}_{XZ}^{\{1\}}$, and 41 002 gave a $K = 128$ code for $\mathcal{E}_{YZ}^{\{1\}}$.

B. Two amplitude damping errors

As determined by exhaustive search in Ref. [19], there are no nontrivial CWS codes capable of detecting the error set $\mathcal{E}^{\{2\}}$ with $n \leqslant 8$. For $n = 9$, the largest CWS code that can detect $\mathcal{E}^{\{2\}}$ has $K = 2$. Interestingly, an exhaustive search of $\mathcal{G}_9$ fails to yield any $K = 2$ codes detecting $\mathcal{E}^{\{2\}}$. However, there are seven elements of $\mathcal{G}_9$ that yield $K = 2$ codes detecting $\mathcal{E}_{XZ}^{\{2\}}$ and 12 elements that yield $K = 2$ codes detecting $\mathcal{E}_{YZ}^{\{2\}}$. For $n = 10$, there are 32 elements of $\mathcal{G}_{10}$ that yield a $K = 2$ code detecting $\mathcal{E}^{\{2\}}$, 309 that yield a $K = 2$ code detecting $\mathcal{E}_{XZ}^{\{2\}}$, and 1327 that yield a $K = 2$ code detecting $\mathcal{E}_{YZ}^{\{2\}}$. There are no larger standard form $n = 10$ codes detecting $\mathcal{E}^{\{2\}}$, $\mathcal{E}_{XZ}^{\{2\}}$, or $\mathcal{E}_{YZ}^{\{2\}}$.

$\mathcal{E}_{YZ}^{\{1\}}, K = 68$

J??H'eKRE_?   J??OQKeeFo?   J??arATbRp_   J??y?cIw^r?

J?@@CoM[C{?   J?@@CoUBf@_   J?@@CoUYK{_   J?A?aKeeBO?

J?C?IKeeFo?   J?C?IKewCo?   J?CarAD'Zp_   J?DAL?w@to_

J?DPTB?BGe_   J?GQXcrr]Z_   J?P@gQ'cPw?   J?Ss[YL[zf_

J@?GYQo@v__   J@KsA?J@u@_   J@P@gQ@_Xw?   JA?hcOo@}P_

JK?HOgoPkR_   JSP@?SECXw?   JTP?OGB?xw?   JW?KA?[Co]?

JW?OOKFeYr?   JWCWoGA_YB?   Jg?w?CRAto?   Jo?OOpFHrp_

$\mathcal{E}_{XZ}^{\{1\}}, K = 68$

J'??C\NJ_{_   J'?CY[kFG^_

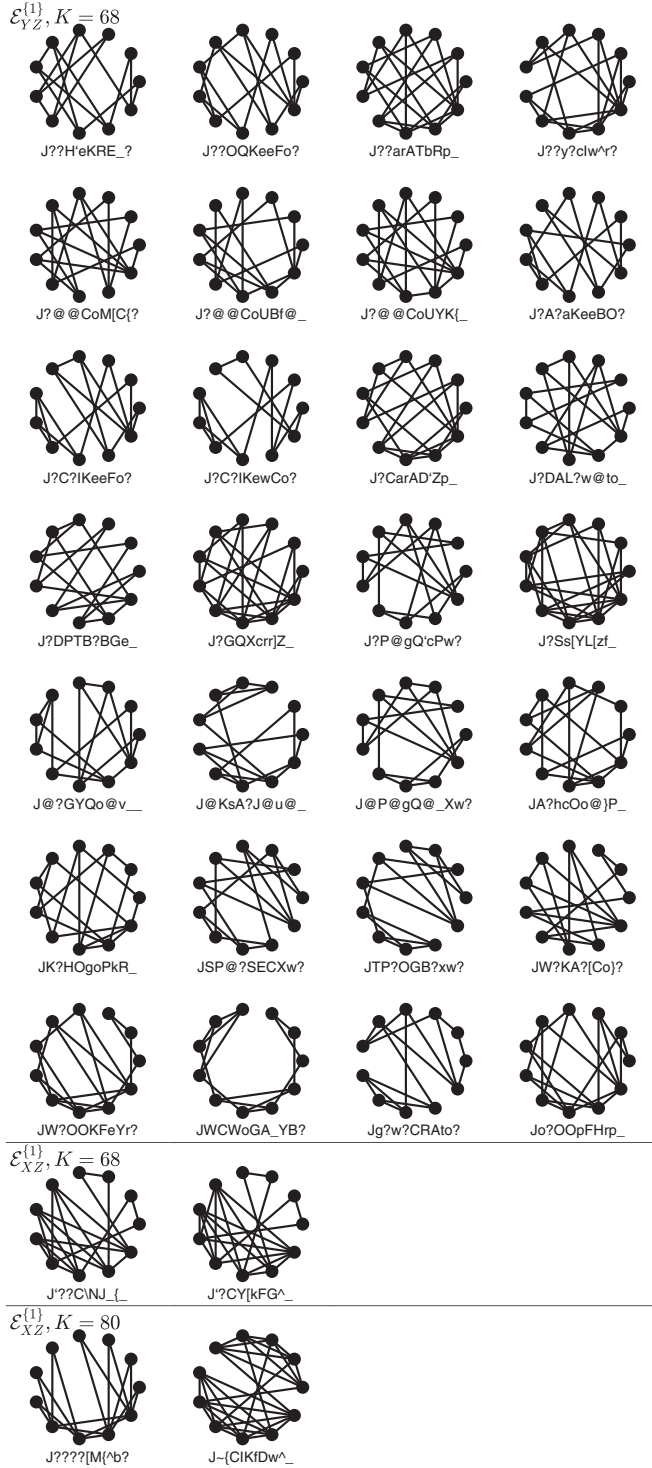$\mathcal{E}_{XZ}^{\{1\}}, K = 80$

J????[M{^b?   J~{CIKfDw^_

FIG. 15. Nonisomorphic graphs yielding ((11,68)) codes detecting $\mathcal{E}_{YZ}^{\{1\}}$, ((11,68)) codes detecting $\mathcal{E}_{XZ}^{\{1\}}$, and ((11,80)) codes detecting $\mathcal{E}_{XZ}^{\{1\}}$.

$\mathcal{E}^{\{2\}}$

J??HGpbcuX_   J?LPCREoxm_

$\mathcal{E}_{XZ}^{\{2\}}$

J?GYdEiT]]?   J?G\Qmy\UX_   J?HN'hXeTH_   J?PTTgxVl\?

JGo_O}ULeF_   JK?{ATPlkL_   JQ?yCSXWkL_   JcG[?~aXjq_

FIG. 16. Nonisomorphic graphs yielding ((11,4)) codes detecting either $\mathcal{E}^{\{2\}}$ or $\mathcal{E}_{XZ}^{\{2\}}$.

algorithm instances yielding codes with these parameters is shown in Table VIII. Note that nearly all of these codes are stabilizer codes. For $n \leqslant 13$, we have used an exact clique finder, whereas for $n = 14$ we have used PLS. The $n = 11$ codes are interesting due to how difficult they are to find. The two graphs found for $\mathcal{E}^{\{2\}}$ are nonisomorphic and eight of those found for $\mathcal{E}_{XZ}^{\{2\}}$ are nonisomorphic. These graphs are shown in Fig. 16. It is easy to find graphs giving codes with $K = 4$ codes for $n = 12$, $K = 8$ for $n = 13$, and $K = 16$ for $n = 14$ (they can be found quickly even with a simple random search). However, to the best of our knowledge, no stabilizer codes with these parameters have been previously published. Furthermore, they are all larger than an optimal $d = 5$ stabilizer code that can correct two arbitrary errors. As such, we include graphs yielding codes of these sizes for $\mathcal{E}^{\{2\}}$, $\mathcal{E}_{XZ}^{\{2\}}$, and $\mathcal{E}_{YZ}^{\{2\}}$ in Fig. 17.

$\mathcal{E}^{\{2\}}$ $n = 12$ $n = 13$ $n = 14$

KCOw?C'GcAWA   LpS_W_J?_A_C?B   Mmln?CO@GO_M?Z?U_

$\mathcal{E}_{XZ}^{\{2\}}$

KrDGpAA@OB?R   LsOXJ?G@GCAL?U   Mp]R?GG?_A_F?K?P_

$\mathcal{E}_{YZ}^{\{2\}}$

KQlw?CbCuDWI   L{KYGoEGm??L?V   MQMiwA@?XAaKoEqD?

FIG. 17. Graphs yielding ((12,4)), ((13,8)), or ((14,16)) codes detecting one of $\mathcal{E}^{\{2\}}$, $\mathcal{E}_{YZ}^{\{2\}}$, or $\mathcal{E}_{XZ}^{\{2\}}$.

As in the single-error-correcting case, any exhaustive search of $\mathcal{G}_n$ for $n \geqslant 11$ is impractical. For $11 \leqslant n \leqslant 14$, we have run 50 000 instances of the genetic algorithm outlined in Sec. III C for each of the three error sets $\mathcal{E}^{\{2\}}$, $\mathcal{E}_{XZ}^{\{2\}}$, and $\mathcal{E}_{YZ}^{\{2\}}$. The best codes found have $K = 4$ for $n = 11$ and 12, $K = 8$ for $n = 13$, and $K = 16$ for $n = 14$. The number of genetic
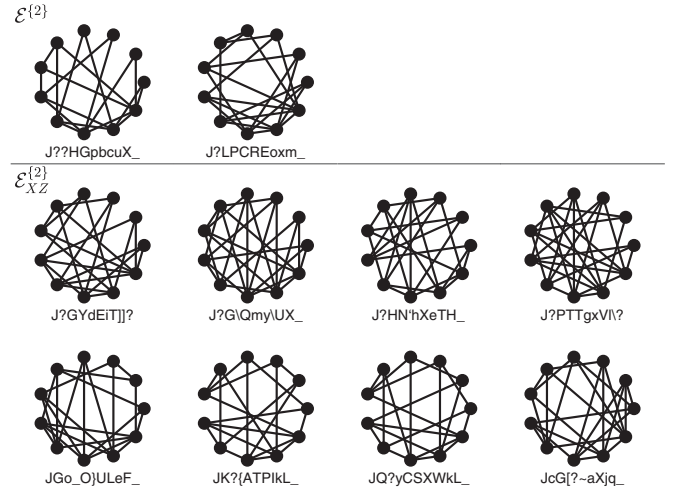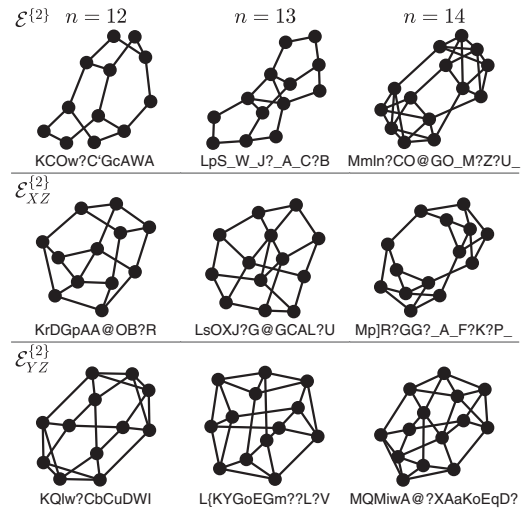
TABLE VIII. The number of genetic algorithm instances out of the 50 000 run that yielded an $((n, K))$ code detecting the given error set.

	$\mathcal{E}^{\{2\}}$	$\mathcal{E}^{\{2\}}_{XZ}$	$\mathcal{E}^{\{2\}}_{YZ}$
$((11,4))$	2	14	0
$((12,4))$	45 912	36 275	43 225
$((13,8))$	38 475	33 163	44 151
$((14,16))$	3 467	5 840	13 148

V. CONCLUSION

We have demonstrated the effectiveness of a number of heuristic approaches to the construction of CWS codes. We have shown that using an approximate maximum clique finding algorithm makes finding larger codes practical. In particular, this has allowed us to find $((9, 97 \leqslant K \leqslant 100, 2))$ and $((11, 387 \leqslant K \leqslant 416, 2))$ codes that are larger than the best known nonadditive codes. We have demonstrated a clustering of clique graph orders and shown a relationship between clique graph order and code size. Furthermore, we have shown that graphs yielding large clique graphs can be found using a genetic algorithm with a crossover operation based on spectral bisection. This search strategy has yielded $((13, 18, 4))$ and $((13, 20, 4))$ codes, which are larger than any previously known code. Finally, we have shown that good codes correcting amplitude damping errors can be found by considering standard form codes that detect one of only three of the 3^n possible LC-equivalent error sets. Coupling this with the genetic algorithm approach, we have found $((11, 68))$ and $((11, 80))$ codes capable of correcting a single amplitude damping error. We have also found $((11, 4))$, $((12, 4))$, $((13, 8))$, and $((14, 16))$ stabilizer codes capable of correcting two amplitude damping errors.

[1] E. Knill and R. Laflamme, Phys. Rev. A **55**, 900 (1997); Phys. Rev. Lett. **84**, 2525 (2000).

[2] D. E. Gottesman, Ph.D. thesis, California Institute of Technology, 1997, arXiv:quant-ph/9705052.

[3] E. M. Rains, R. H. Hardin, P. W. Shor, and N. J. A. Sloane, Phys. Rev. Lett. **79**, 953 (1997).

[4] S. Yu, Q. Chen, C. H. Lai, and C. H. Oh, Phys. Rev. Lett. **101**, 090501 (2008).

[5] S. Yu, Q. Chen, and C. H. Oh, arXiv:0709.1780.

[6] E. M. Rains, IEEE Trans. Inf. Theory **45**, 266 (1999).

[7] J. A. Smolin, G. Smith, and S. Wehner, Phys. Rev. Lett. **99**, 130505 (2007).

[8] M. Grassl and M. Rotteler, in *Proceedings of the 2008 IEEE International Symposium on Information Theory, Toronto, Canada* (IEEE, Piscataway, NJ, 2008), pp. 300–304.

[9] A. Cross, G. Smith, J. A. Smolin, and B. Zeng, IEEE Trans. Inf. Theory **55**, 433 (2009).

[10] I. Chuang, A. Cross, G. Smith, J. Smolin, and B. Zeng, J. Math. Phys. **50**, 042109 (2009).

[11] M. Van den Nest, J. Dehaene, and B. De Moor, Phys. Rev. A **69**, 022316 (2004).

[12] M. Grassl, A. Klappenecker, and M. Rotteler, in *Proceedings of the IEEE International Symposium on Information Theory, Lausanne, Switzerland* (IEEE, Piscataway, NJ, 2002), p. 45.

[13] D. Schlingemann, Quantum Inf. Comput. **2**, 307 (2002).

[14] L. E. Danielsen, Master's thesis, The University of Bergen, 2005, arXiv:quant-ph/0503236.

[15] L. E. Danielsen and M. G. Parker, J. Comb. Theory, Ser. A **113**, 1351 (2006).

[16] L. E. Danielsen, Database of self-dual quantum codes, online, available at www.ii.uib.no/~larsed/vncorbits/.

[17] R. M. Karp, in *Complexity of Computer Computations* (Springer, Boston, 1972), pp. 85–103.

[18] W. Pullan, J. Combin. Optim. **12**, 303 (2006).

[19] T. Jackson, M. Grassl, and B. Zeng, in *Proceedings of the 2016 IEEE International Symposium on Information Theory, Barcelona, Spain* (IEEE, Piscataway, NJ, 2016), pp. 2264–2268.

[20] B. D. McKay and A. Piperno, J. Symbolic Comput. **60**, 94 (2014).

[21] B. D. McKay and A. Piperno, Nauty and Traces user's guide (Version 2.5).

[22] F. Harary and E. M. Palmer, *Graphical Enumeration* (Academic Press, New York, 1973).

[23] Q. Wu and J.-K. Hao, Eur. J. Operat. Res. **242**, 693 (2015).

[24] M. R. Garey and D. S. Johnson, *Computers and Intractability* (W. H. Freeman, New York, 1979).

[25] K. M. Hall, Manage. Sci. **17**, 219 (1970).

[26] W. E. Donath and A. J. Hoffman, IBM Technical Disclosure Bulletin **15**(3), 938 (1972).

[27] M. Fiedler, Czech. Math. J. **25**, 619 (1975).

[28] A. Pothen, H. D. Simon, and K.-P. Liou, SIAM J. Matrix Anal. Appl. **11**, 430 (1990).

[29] M. Fiedler, Czech. Math. J. **23**, 298 (1973).

[30] D. Whitley, Stat. Comput. **4**, 65 (1994).

[31] S. Luke, *Essentials of Metaheuristics* (Lulu, Morrisville, NC, 2013).

[32] S. Legg, M. Hutter, and A. Kumar, in *Proceedings of the 2004 Congress on Evolutionary Computation, Portland, Oregon* (IEEE, Piscataway, NJ, 2004), pp. 2144–2151.

[33] A. Globus, J. Lawton, and T. Wipke, Nanotechnology **10**, 290 (1999).

[34] S. Stone, B. Pillmore, and W. Cyre (unpublished).

[35] K. Kraus, *States, Effects and Operations: Fundamental Notions of Quantum Theory* (Springer, Berlin, 1983).

[36] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge University Press, New York, 2011).

[37] M. Hein, W. Dür, J. Eisert, R. Raussendorf, M. Van den Nest, and H.-J. Briegel, in *Quantum Computers, Algorithms and Chaos* (IOS Press, Amsterdam, 2005), pp. 115–218.

[38] S. Y. Looi, L. Yu, V. Gheorghiu, and R. B. Griffiths, Phys. Rev. A **78**, 042303 (2008).

[39] Z. Ji, J. Chen, Z. Wei, and M. Ying, Quantum Inf. Comput. **10**, 97 (2010).

[40] Y. Li, I. Dumer, M. Grassl, and L. P. Pryadko, Phys. Rev. A **81**, 052337 (2010).

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

[42] E. M. Rains, IEEE Trans. Inf. Theory **45**, 2489 (1999).

[43] G. Nebe, E. M. Rains, and N. J. A. Sloane, *Self-Dual Codes and Invariant Theory* (Springer, Berlin, 2006).

[44] M. Grassl, Bounds on the minimum distance of linear codes and quantum codes, online, available at www.codetables.de.

[45] J. Löfberg, in *Proceedings of the 2004 IEEE International Conference on Robotics and Automation, New Orleans, LA* (IEEE, Piscataway, NJ, 2004), pp. 284–289.

[46] J. Bierbrauer, R. Fears, S. Marcugini, and F. Pambianco, IEEE Trans. Inf. Theory **57**, 4788 (2011).

[47] J. Konc and D. Janezic, Match. Commun. Math. Comput. Chem. **58**, 569 (2007).

[48] W.-T. Yen and L.-Y. Hsu, arXiv:0901.1353.

[49] See Supplemental Material at http://link.aps.org/supplemental/10.1103/PhysRevA.100.062303 for the graph and associated classical code for all the CWS codes.

[50] T. M. Fruchterman and E. M. Reingold, Software: Practice Exper. **21**, 1129 (1991).

[51] M. Grassl, Z. Wei, Z.-Q. Yin, and B. Zeng, in *Proceedings of the 2014 IEEE International Symposium on Information Theory Honolulu, HI* (IEEE, Piscataway, NJ, 2014), pp. 906–910; M. Grassl, L. Kong, Z. Wei, Z.-Q. Yin, and B. Zeng, IEEE Trans. Inf. Theory **64**, 4674 (2018).

[52] A. S. Fletcher, P. W. Shor, and M. Z. Win, IEEE Trans. Inf. Theory **54**, 5705 (2008).

[53] P. W. Shor, G. Smith, J. A. Smolin, and B. Zeng, IEEE Trans. Inf. Theory **57**, 7180 (2011).

[54] R. Lang and P. W. Shor, arXiv:0712.2586.