High-Rate Storage Codes on Triangle-Free Graphs

Alexander Barg, *Fellow, IEEE*, and Gilles Zémor

***Abstract*— Consider an assignment of bits to the vertices of a connected graph $G(V,E)$ with the property that the value of each vertex is a function of the values of its neighbors. A collection of such assignments is called a *storage code* of length $|V|$ on G. The storage code problem can be equivalently formulated as maximizing the probability of success in a *guessing game* on graphs, or constructing *index codes* of small rate. If G contains many cliques, it is easy to construct codes of rate close to 1, so a natural problem is to construct high-rate codes on triangle-free graphs, where constructing codes of rate $>1/2$ is a nontrivial task, with few known results. In this work we construct infinite families of linear storage codes with high rate relying on coset graphs of binary linear codes. We also derive necessary conditions for such codes to have high rate, and even rate potentially close to one. We also address correction of multiple erasures in the codeword, deriving recovery guarantees based on expansion properties of the graph. Finally, we point out connections between linear storage codes and quantum CSS codes, a link to bootstrap percolation and contagion spread in graphs, and formulate a number of open problems.**

***Index Terms*— Storage codes, index codes, guessing number, Cayley graphs, CSS codes, bootstrap percolation.**

I. Introduction

IN THIS paper we consider a class of codes on graphs known as storage codes. Given an undirected graph $G(V,E)$ with N vertices, denote by $\mathcal{N}(v)=\{u:(v,u)\in E\}$ the set of neighbors of the vertex v in G. Consider a set of vectors $\mathcal{C}=\{x : x\in Q^N\}$ over a finite alphabet Q, where the coordinates are indexed by the vertices in V. We assume some fixed order of the vertices of G throughout the paper. The set $\mathcal{C}$ is said to form a *storage code* if for every $v\in V$ and $x,y\in\mathcal{C}$, if $x_u=y_u$ for all $u\in\mathcal{N}(v)$ then also $x_v=y_v$. In other words, the codewords are written on the vertices of G, and for any codeword the value of the vertex v can be uniquely determined by the values of its neighbors. Thinking of storing the coordinates of the codeword at different nodes of a distributed storage system, this definition implies that an erased coordinate (vertex) can be recovered in a local way from its immediate neighborhood. This definition can be also phrased in terms of recovery functions f_v that map the subcodewords (x_u), $u\in\mathcal{N}(v)$, on the value x_v of v. Note that f_v generally depends on v, and the functions for different vertices may be different.

The concept of storage codes on graphs was introduced around 2014 by Mazumdar [22], [23] and Shanmugam and Dimakis [27], motivated by earlier works on index coding by Alon *et al.* [1] and codes with locality for distributed storage introduced by Gopalan *et al.* [16]. The defining property of codes with locality is the ability to recover a missing coordinate of the codeword based on the values of the symbols in a small subset of other coordinates (the repair group), with the goal of minimizing the size of these subsets and reducing internodal communication for completing the recovery task. Storage codes on graphs additionally assume that the links between the nodes are established based on physical proximity and the associated energy constraints, limitations of the system architecture, or other features with the same effect. This constraint, modeled by the graph G, defines the neighborhood of each vertex and guides the construction of the code used to store information on the vertices. Similarly, the problem of index coding addresses constructions of broadcast functions that distribute information to the vertices of the graph whereby each vertex has access to the "side information" stored on the vertices in its neighborhood in the graph.

Essentially the same problem, in a different guise, appears in a line of works devoted to guessing games on graphs, e.g., [8], [10], which has developed independently of both the storage codes and index coding problems. That these groups of problems are largely equivalent was realized in a number of papers, and the historical development is detailed in [2] from the index codes' perspective. We present a brief summary of the relations between these three problems in Section II-A below.

Example 1.1: To give an example, consider the graph in Figure 1. We can construct a storage code by assigning to its vertices $1,2,\dots,5$, binary vectors $(x_1,x_2,\dots,x_5)$ that satisfy $x_1=x_2$ and $x_3+x_4+x_5=0$. Clearly, every vertex can recover its value from its neighborhood, for instance, the recovery functions of the vertices 1 and 3 are $f_1(\{x_2,x_3\})=x_2$ and $f_3(\{x_1,x_4,x_5\})=x_4+x_5$, respectively. All the possible assignments give rise to a linear binary storage code of length $N=5$ and dimension 3, so the rate of the code is $R(G)=3/5$.

The main question concerning storage codes is determining the largest possible cardinality of the code $\mathcal{C}$ for a given graph G. Below we denote by $R_q(G)$ the maximum possible

Manuscript received 5 October 2021; revised 25 March 2022; accepted 7 July 2022. Date of publication 15 July 2022; date of current version 22 November 2022. The work of Alexander Barg was supported by NSF under Grant CCF2104489 and by NSF-BSF Grant CCF-2110113. The work of Gilles Zémor was supported in part by ANR through Project QUDATA under Grant ANR-18-CE47-0010. This work was presented in part at the Twelfth International Workshop on Coding and Cryptography, March 2022, Rostock, Germany and at the International Conference on Coding and Cryptography, May 2022, Shanghai, China. *(Corresponding author: Alexander Barg.)*

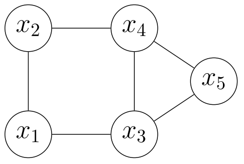
Alexander Barg is with ISR and Department of ECE, University of Maryland, College Park, MD 20742 USA (e-mail: abarg@umd.edu).

Gilles Zémor is with the Mathematics Institute (IMB, CNRS UMR 5251), University of Bordeaux, 33405 Talence, France (e-mail: zemor@math.u-bordeaux.fr).

Communicated by L. Dolecek, Associate Editor for Coding and Decoding.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TIT.2022.3191309.

Digital Object Identifier 10.1109/TIT.2022.3191309



$$x_1 + x_2 = 0$$
$$x_3 + x_4 + x_5 = 0$$

Fig. 1. The graph and storage code in Example 1.1.

rate $\frac{1}{N}\log_q|\mathcal{C}|$ of a q-ary storage code for a given graph G. Determining $R_q(G)$ (or even $R(G) := \sup_q R_q(G)$ if we wish to optimize on the choice of the code alphabet) is a difficult problem related to the so-called minrank of the graph, and we refer to [24] for the best known bounds as well as an overview of the earlier results. Below we limit ourselves to finite field alphabets $Q = \mathbb{F}_q$ and assume that the code $\mathcal{C}$ forms a *linear subspace* of $\mathbb{F}_q^N$. While nonlinear functions can sometimes attain higher rates [1], adding linearity enables one to rely on various known structures to construct families of storage codes. Generally, improved rates associated with nonlinear storage/index codes require specially designed dependence graphs, while linear maps support constructions for large graph families based on universal methods.

In the next section we overview the known constructions of storage codes. It turns out that in dense graphs, for instance, graphs with many cliques, attaining high rate is easy: in particular, if the graph G on vertices $1, 2, \dots, n$ is itself a clique, then it affords a storage code of rate close to one defined by the single parity check equation $\sum_{i=1}^n x_i = 0$, that is common to all the vertices. More generally, if we partition the vertex set of the graph into cliques, we can define a storage code with one single equation for every clique. In the example of Figure 1, there are two equations, one for clique $\{1, 2\}$ and one for clique $\{3, 4, 5\}$. Therefore graphs with many large cliques tend to admit storage codes of high rate. But in graphs with no cliques of size larger than 2, i.e., triangle-free graphs, the simple clique partition strategy achieves at best rate $1/2$. Other more refined constructions also fail to exceed rate $1/2$ when graphs are triangle-free (more on this below), and it was also proved that bipartite graphs, a large class of triangle-free graphs, do not admit storage codes of rates that exceed $1/2$. This motivated us to look for triangle-free graphs that do admit storage codes of rate larger than $1/2$, and this forms the main subject of this paper. Constructing codes of high rate on such graphs represents a challenge, and early studies [10] have conjectured that with no triangles, $R = 1/2$ is the largest attainable rate value. While this conjecture was refuted in [8], only isolated examples of such codes (and their direct extensions to infinite families; see Remark 2.1 below) have been presented in the literature.

The codes that we consider are constructed on Cayley graphs on $\mathbb{F}_2^r$ (coset graphs of binary linear codes). While coset graphs form a classic topic in coding theory, one motivation of this work is drawn from a recent construction of quantum CSS codes obtained from linear spaces defined by adjacency matrices of coset graphs [12]. Our main results are constructions of two infinite families of binary linear storage codes of rate above 1/2, one of which in fact has rate $R_2(G)$ approaching 3/4 for growing length N, exceeding the rate 5/8 of the unique previously known binary storage code with rate above 1/2 for triangle-free graphs. The exact dimension of these storage codes is computed. We also formulate necessary conditions for the code dimension to be high compared to the number of vertices N.

In addition, we present computer-assisted results that give sporadic examples of storage codes with high rate on triangle-free coset graphs: in particular we uncover an example of a storage code on a triangle-free graph whose rate exceeds 3/4 and surpasses that of all previously known storage codes over any alphabet.

Finally, we address a problem left open in [23], namely that of correcting multiple erasures in the codeword: we derive local recovery guarantees based on expansion properties of the graph, and make a connection with *bootstrap percolation*.

The paper is organized as follows. Section II is dedicated to background material on storage codes, with connections to index coding and guessing games. Section III introduces storage codes on coset graphs of linear codes and makes a connection with quantum coding, transforming a known family of quantum codes into storage codes. Section IV derives necessary conditions for storage codes on triangle-free coset graphs to have large rates. Section V is devoted to constructing a family of binary linear codes and proving that their coset graphs are triangle-free and admit storage codes of rate approaching 3/4. Section VI addresses the problem of correcting multiple erasures for storage codes on expander graphs. Finally, Section VII concludes with results on some storage code rates for triangle-free coset graphs found with computer help, and with miscellaneous comments and open problems.

II. Background

A. Storage Codes, Index Codes, and Guessing Games

The problem of finding the largest storage code for the graph G is closely related to two other recently introduced problems in computer science, *index coding* and *guessing games on graphs*. In the symmetric index coding problem, an information vector $x = (x_1, \dots, x_N)$ is to be distributed to N users via a single broadcast with the goal of furnishing user i with the symbol x_i. The users possess "side information" about their intended messages in the form of a subset of coordinates of the vector x. Specifically, let $N_i \subset \{1, \dots, N\}$ be a subset of indices, and suppose that user i has access to symbols $x(N_i) := (x_j, j \in N_i)$. The index coding problem seeks to construct an encoding (broadcast) function $f : \mathbb{F}_q^N \to W$, where $K := \log_q|W| < N$, and N decoding functions ϕ_i such that $\phi_i(f(x), x(N_i)) = x_i$ for all $i = 1, \dots, N$.

The index coding problem is conveniently described in terms of a *side information graph* $G(V, E)$ with N vertices and $(i, j) \in E$ if and only if $j \in N_i$. We assume that the relation $j \in N_i$ is symmetric in i, j, and thus the edges are undirected. The value $I_q(G, f) := K/N$ is called the rate of the q-ary symmetric index code $(f, \{\phi_i\})$, and the objective of the construction is to devise broadcast functions (index codes) of minimum rate for a given side information graph.

The minimum possible rate of the index code for the side information graph G is called the capacity $I_q(G)$ of index coding for G. It is often possible to reduce the rate value $I_q(G)$ by increasing q, resulting in a universal characterization $I(G)$ of index coding for a given graph.

The problems of storage coding and index coding are to a large extent equivalent, namely, the following is true:

(A) Any graph G defines an index coding problem $(f, \{\phi_i\})$ wherein the sets $N_i = \mathcal{N}(i)$ are the neighborhoods of vertices $i \in V$. Furthermore, for any value s of the broadcast function, the preimage $f^{-1}(s)$ is a storage code for the graph G.

(B) If there is a partition $(\mathcal{C}_j)_{j\in W}$ of $\mathbb{F}_q^N$ into a set of storage codes $\mathcal{C}_s$ for G, then the map $\mathbb{F}_q^N \to W$ that associates a vector x to the index s such that $x \in \mathcal{C}_s$, is a valid broadcast function for the index coding problem with side information graph G.

(C) Moreover, if the broadcast function f for an index coding problem defined by a graph G is linear, then its kernel is a storage code. Conversely, if $\mathcal{C}$ is an $\mathbb{F}_q$-linear $[N, K]$ storage code, then any syndrome function $f : \mathcal{C} \to \mathbb{F}^{N-K}$ is a broadcast function for the corresponding index coding problem. This link is a starting point of the equivalence proof of the two problems in [22].

As shown in [1], [23]

$$1 - I_q(G) \le R_q(G) \le 1 - I_q(G) + N^{-1}\log_q(N \ln q),$$

and thus $\sup_q(R_q(G)) + \inf_q(I_q(G)) = 1$.

Another equivalent formulation of the storage coding problem arises from the study of guessing games on graphs initiated in [25]. In one version of the game, the vertices in V are assigned elements of a finite set Q (colors), and each vertex v attempts to guess its color based on the colors of its neighbors in $\mathcal{N}(v)$. The game is won if all the vertices correctly guess their colors. They may agree on the guessing strategy in advance, but once they are assigned colors, all communication is forbidden. The assignment $x \in Q^N$ is assumed uniformly random, and the participants (vertices) attempt to maximize their probability of success P_s. Any storage code $\mathcal{C}(G)$ for the graph G defines a guessing strategy, namely every participant will assume that $x \in \mathcal{C}$ and reconstruct their x_v from the values of their neighbors: we thus have $P_s(\mathcal{C}, G) = \frac{|\mathcal{C}(G)|}{q^N} = \exp(\log|\mathcal{C}| - N)$. Conversely, any guessing strategy defines a storage code, namely the set of values x for which the strategy succeeds, and thus maximizing P_s is equivalent to constructing large-size storage codes. The authors of [8] define the *guessing number* of G as $\mathsf{gn}_q(G) = N + \log_q \max P_s(\mathcal{C}, G)$; thus in our notation $\mathsf{gn}_q(G) = NR_q(G)$. In the context of guessing games it has been shown that the storage capacity $R_q(G)$ is almost monotone on q. Namely, the following is true.

Theorem 1 ([8], [10], [14]): For any graph G, alphabet size q, and $\epsilon > 0$ there exists $q_0(G, q, \epsilon)$ such that for all $q' > q_0$

$$R_{q'}(G) \ge R_q(G) - \epsilon.$$

Consequently, the limit $\lim_{q\to\infty} R_q(G)$ exists.

The equivalence between index coding, storage coding, and guessing games is further discussed in [2].

B. Constructions of Storage Codes

To motivate the problem that we will study in the next section, we briefly overview the known constructions; see [8], [10], [24] and also [3, Ch.6] for the details.

1) Matching Construction: A matching $M \subset E$ in a graph $G(V, E)$ is a subset of edges such that every vertex $v \in V$ is incident to at most on edge from M, and a matching is *perfect* if every $v \in V$ is incident to exactly one edge from it. A matching M defines a linear storage code $\mathcal{C}\colon \mathbb{F}_q^{|M|} \to \mathbb{F}_q^N$ by associating to $(x_e)_{e\in M}$ the vector $c \in \mathbb{F}_q^N$ such that for every vertex v incident to an edge $e \in M$ we assign $c_v = x_e$ and for every remaining vertex v we put $c_v = 0$. In particular, if G is d-regular and bipartite, it contains a perfect matching, giving rise to a storage code $\mathcal{C}(G)$ of rate $1/2$ independently of q. If G is not bipartite, we can take a double cover[1] $\bar{G}$ of G. Now, $\bar{G}$ is a bipartite regular graph, and we can apply the matching encoding and obtain a code $\bar{\mathcal{C}} \subset \mathbb{F}_q^{2N}$. This code can be mapped on a code $\mathcal{C}$ over $\mathbb{F}_{q^2}^N$ obtained by grouping together pairs of symbols indexed by vertices $v', v'' \in V(\bar{G})$ that correspond to the same vertex $v \in V(G)$. This construction gives a way of obtaining a code of rate $1/2$ on any regular graph. Clearly, the value of any vertex in the codeword is recoverable from its neighbors, implying that $\mathcal{C}$ is a storage code for G.

2) Edge-Vertex Construction: An alternative way of obtaining a code of rate $1/2$ on a d-regular graph $G(V, E)$ is the following. Consider the space $(\mathbb{F}_q)^{|E|}$ of vectors indexed by the edges of G. Next, map this space on $(\mathbb{F}_q^d)^N \cong \mathbb{F}_{q^d}^N$ by assigning to each vertex v a d-vector formed of the symbols written on the edges incident to it. In other words, $\mathcal{C}(G) = \{(c_v, v \in V)\}$ is obtained as the image of the mapping

$$\begin{aligned} &\mathbb{F}_q^{|E|} \to \mathbb{F}_{q^d}^N \\ &(x_e)_{e\in E} \mapsto (c_v)_{v\in V}, \text{ where } c_v = (x_e)_{e\in\partial(v)}, \end{aligned} \qquad (1)$$

where $\partial(v)$ is the edge neighborhood of v in G. Since $|(\mathbb{F}_q)^{|E|}| = q^{\frac{dN}{2}}$, the rate of $\mathcal{C}$ is indeed 1/2. In Figure 2 we show this construction for the cycle C_5, where we first place symbols of the q-ary alphabet on the edges, and then assign to each vertex the symbols on the edges that are incident to it. Now each vertex can recover its pair of symbols from its neighbors: for instance, v_1 takes the second symbol from v_2 and the first one from v_5 (note again that we fix the order of the vertices). This gives a code of size $|\mathcal{C}| = q^{|E|}$ with $|E| = 5$, and thus the rate is $R(\mathcal{C}) = \frac{1}{N}\log_{q^2}|C| = 1/2$.

3) Clique-Vertex Construction: The edge-vertex construction affords a straightforward generalization if every vertex $v \in V$ is incident to the same number (say m) of k-cliques, where $k > 2$. Let $\mathscr{K}$ be the set of k-cliques in G and let us map $(\mathbb{F}_q^{k-1})^{|\mathscr{K}|}$ to $(\mathbb{F}_q^m)^n$ by placing a q-ary code of length k and dimension $k-1$ on every clique and distributing the symbols of the k-codeword to the vertices that form the clique.

[1]A bipartite double cover of G is a graph with adjacency matrix $\begin{pmatrix} 0 & A \\ A & 0 \end{pmatrix}$, where A is the adjacency matrix of G.

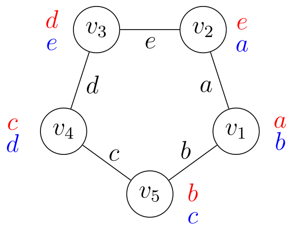


Fig. 2. Edge-vertex construction of a storage code.

The rate of the code $\mathcal{C}(G)$ obtained as a result equals $(k-1)/k$; for instance, a triangular lattice gives rise to a code of rate $2/3$, etc.

4) Fractional Clique Cover: A fractional clique cover of a graph G is a collection $\mathscr{K}$ of cliques κ together with a weight $w_\kappa \in [0,1]$, such that for every vertex v, we have

$$\sum_{\kappa\in\mathscr{K}:v\in\kappa} w_\kappa \ge 1 \tag{2}$$

To construct a storage code for G, we apply the clique-vertex construction to the subset of cliques that make up the fractional clique cover. The vertices may not all be incident to the same number of cliques, in which case the alphabet size may depend on the vertex (a mixed-alphabet code), or we have to use the alphabet $\mathbb{F}_q^m$ where m is the largest vertex-degree in the vertex to clique incidence graph.

C. Bounds for Storage Codes

It is known that $R_q(G)$ satisfies the constraints

$$M(G) \le NR_q(G) \le N - \alpha(G) \tag{3}$$

where $M(G)$ is the size of the largest matching in G and $\alpha(G)$ is the size of the largest independent set in G, i.e., vertices with no edge among them. This result was proved in [23] for storage codes and in an earlier independent work [10] in the language of guessing games. Since the complement of a maximum independent set forms a vertex cover in G (a set of vertices that touches every edge in $E(G)$), the upper bound in (3) is often stated in terms of the vertex cover number. A consequence of (3) is that if the graph G is not only triangle-free but bipartite, we must have $R \le 1/2$ since a bipartite graph has an independent set of size at least $N/2$.

To state another bound, recall that a fractional clique cover is called *regular* if its weighting $w: K(G) \to [0,1]$ satisfies (2) with equality. As shown in [10] (for the guessing number) and in [5] (for the broadcast rate),

$$R(G) \ge 1 - \frac{1}{N}\kappa_f(G),$$

where $\kappa_f(G) := \min \sum_\kappa w_\kappa$ is the minimum weight of a regular fractional cover. The authors of [10] observed that this bound holds with equality for perfect graphs and cycles or their complements.

A linear program for bounding $R_q(G)$ was introduced by the authors of [24], adapting a similar bound for $I_q(G)$ [5] to the case of storage codes. As a result, [24] showed an upper bound $R_q(G) \le 1/2$ for several classes of graphs.

Below we focus on the question of identifying graphs that support storage codes of the largest rate. Without any constraints, the complete graph K_N affords a linear storage code of rate $\frac{N-1}{N}$ (defined by the single equation $\sum_{v\in V} x_v = 0$). The problem becomes less trivial if we limit ourselves to sparse graphs G, for instance, d-regular graphs on N vertices with varying N.

Proposition 2: Let G be a connected d-regular graph on N vertices, then $R_q(G) \le \frac{N+1}{N} - \frac{1}{d+1}$. There exist d-regular graphs with $R_q(G) \ge 1 - \frac{1}{d-2}$.

Proof: Recall that the Cartesian product of graphs G and H is a graph with the set of vertices $V(G)\times V(H)$ in which vertices (u,u') and (v,v') are adjacent if and only if either $u = v$ and $(u',v') \in E(H)$ or $(u,v)\in E(G)$ and $v = v'$. Take $s \ge 3$ and let G be the Cartesian product of K_{d-2} and a cycle of length s. According to the above definition, the graph obtained as a result is formed of s copies of the graph G in which copies of the same vertex are connected by the edges of the cycle. Now put a code of length $d-2$ on each clique independently, obtaining the claimed lower bound. On the other hand, any d-regular graph on N vertices contains an independent set of size $\lfloor \frac{N}{d+1} \rfloor$ [26], and together with (3) this implies the upper bound. □

Remark 2.1: This last proposition highlights the following fact. If we have a graph G that admits a storage code C of rate $R_q(G)$, we can construct many more graphs with the same storage rate, simply by taking $k \ge 1$ disjoint copies of the original graph G. The associated storage code will consist of the Cartesian product of k copies of the original code C, whose codewords consist therefore of k successive codewords of C. We can also add arbitrary edges to and between the copies of G that the storage code will simply ignore. In what follows, we will be looking for the highest possible rate of a storage code for graphs with specific properties, and will conflate a code with its successive Cartesian powers, considering that we are dealing with the same code 'up to repetitions'.

III. Storage Codes on Triangle-Free Graphs and the CSS Code Connection

The graphs used to obtain storage code rates close to one are dense and contain a large number of cliques. It is therefore natural to consider the question of the largest attainable rate for graphs that contain no cliques, i.e., triangle-free graphs. For such graphs none of the methods mentioned above yield a storage code rate in excess of $1/2$; in particular, as noted in [5] (in the language of index codes), $\kappa_f(G) \ge N/2$. The same observation was also made in [8] in terms of the guessing number of triangle-free graphs. Both [5] and [8] reported only one example of a storage code on triangle-free graphs for which $R_2(G) > \frac{1}{2}$ (up to repetitions, in the sense of Remark 2.1). This code is associated to the graph Γ shown below in Fig. 3 (the authors of [8] also gave several examples of triangle-free graphs with $R_q(G) > \frac{1}{2}$ for $q > 2$). This graph can be defined in several ways and is known as the *Clebsch graph* or a *folded cube*, see Figure 3. The storage code constructed on Γ is a linear binary code $\mathcal{C}$ of length

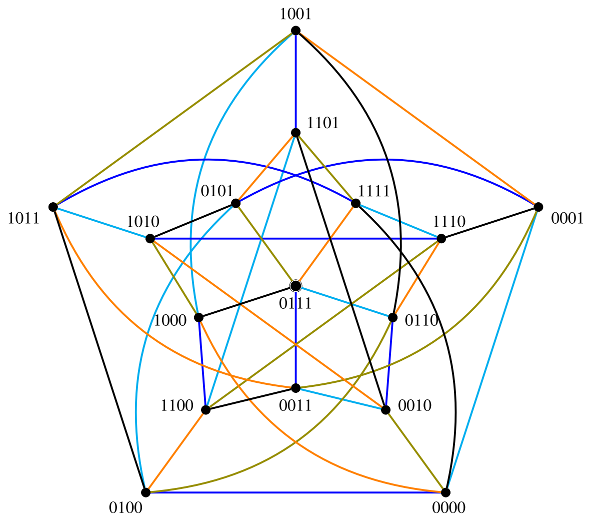


Fig. 3. The Clebsch graph Γ (a folded cube): A 5-regular triangle-free graph on $n = 16$ vertices obtained by identifying each pair of opposite vertices in the 5-dimensional hypercube. It is also a coset graph of the binary code $\{00000, 11111\}$.

$N = 16$ whose parity-check matrix has the form

$$\tilde{A} = I_N + A(\Gamma), \tag{4}$$

where $A(\Gamma)$ is the adjacency matrix of Γ. The identity matrix I_N is added since A includes only the neighborhood $\mathcal{N}(v)$ but not the vertex v itself. Upon checking that the $\mathbb{F}_2$-rank of $\tilde{A}$ equals 6, we conclude that the dimension of the code equals $N - \operatorname{rk} \tilde{A} = 10$, so $R_2(G) \geq \frac{5}{8}$. This example refuted a conjecture in [10] which suggested that the fractional clique cover bound holds with equality for triangle-free graphs. It is also easy to check that Γ contains independent sets of size 5, and thus from (3) we conclude that $\frac{5}{8} \leq R_2(\Gamma) \leq \frac{11}{16}$, where the upper bound is sharper than the result of Prop. 2. Using the terminology of guessing games, the guessing number of the graph Γ satisfies $10 \leq \mathsf{gn}_2(\Gamma) \leq 11$, or rephrasing again, the side information graph Γ affords a binary index code of rate $3/8$.

Note that the recovery functions of the vertices use full neighborhoods (the parity-check equations have weight 6), even though the general definition of the storage code does not include this constraint. We call binary codes from this subclass *full-parity* storage codes, and it is only such codes that we consider in this and the next sections.

The Clebsch graph forms a rather special example: it is a unique strongly regular graph with the parameters $(16, 5, 0, 2)$; see [15], Theorem 10.6.4. Six other triangle-free strongly regular graphs are known [6, p. 119], and all the other examples of (nonbinary) storage codes of rate greater than $\frac{1}{2}$ in [8] are drawn from this list.

A. Storage Codes on Cayley Graphs

Recall that the Cayley graph $\mathsf{Cay}(\mathscr{G}, S)$ of the group $\mathscr{G}$ for a given set of generators S has $\mathscr{G}$ as its set of vertices, and the vertices g_1 and g_2 are connected by an edge if there is an element $s \in S$ such that $g_1 s = g_2$. For the group $\mathscr{G} = \mathbb{F}_2^r$, which is the only group we will consider, any subset S coincides with its inverse S^{-1}, so the graphs that we study are undirected. Since the generators are r-dimensional binary vectors, we also write the group action additively. Write the elements of S as columns of an $r \times n$ matrix H and consider the binary code C defined by H as the parity-check matrix. The graph $\mathsf{Cay}(\mathscr{G}, S)$ can be also viewed as a *coset graph* of the code C constructed on the set of cosets in $\mathbb{F}_2^n/C$, wherein two cosets are connected with an edge if and only if the Hamming distance between them (as subsets) is one.

Observe that the Clebsch graph Γ can be viewed as a Cayley graph over the group $\mathbb{F}_2^4$. Namely, consider the set S whose elements form the matrix

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}. \tag{5}$$

Indeed, as easily checked, the canonical generators e_1, e_2, e_3, e_4 connect a vertex $x \in \mathbb{F}_2^4$ to its neighbors at Hamming distance one, while their sum $e_1 + \cdots + e_4$ connects it to its opposite vertex. See Fig. 3 for one possible vertex labeling, where each color corresponds to the action of a specific generator. Viewing the matrix H as a parity-check matrix of a $[5, 1]$ binary repetition code, we see that Γ is in fact the *coset graph* of this code.

Motivated by this example, we now investigate other Cayley graphs over binary groups $\mathbb{F}_2^r$ and exhibit more exceptional graph families that admit binary storage codes of rate greater than $1/2$. These graphs have connections to both classical and quantum coding theory.

1) Notation: Our notation is as follows. We are given a binary code C with parameters $[n, k, d]$ with a fixed parity-check matrix H which we are free to choose. Next we construct the coset graph $G = \mathsf{Cay}(\mathbb{F}_2^r, S)$ where $r = n - k$ is the number of rows and S is the set of columns of H. Note that G is a regular graph of degree $|S| = n$. The adjacency matrix A of this graph is symmetric, of order $N = 2^r$, and its rows and columns are labeled by the vectors $x \in \mathbb{F}_2^r$: we would like row x to specifiy the parity-check equation that recovers the value of the vertex x from its neighbors. Note however that $A_{x,x} = 0$, therefore, to involve the value supported by x in the parity-check equation, we consider, as in (4), the matrix

$$\tilde{A} = I_N + A(G). \tag{6}$$

Our storage code is finally a linear space in $\mathbb{F}_2^N$ defined as $\mathcal{C} = \ker(\tilde{A})$.[2] Note that the matrix $\tilde{A}$ is the adjacency matrix of the graph $\mathsf{Cay}(\mathbb{F}_2^r, S)$ to which we have added self-loops at every vertex, or equivalently the Cayley graph $\mathsf{Cay}(\mathbb{F}_2^r, S')$, where S' is obtained from S by adding 0 to the set of generators. The main problem addressed below is analyzing the dimension of $\mathcal{C}$ both in general and for several specific constructions.

As our first observation, note that once the minimum distance of the code C is at least 4, then so is the girth of the graph $\mathsf{Cay}(\mathbb{F}_2^r, S)$, i.e., the graph is triangle-free. We will therefore assume that all the small codes below have distance 4 or more.

[2]Notice that we deal with two types of binary codes: the codes in $\mathbb{F}_2^n$ and codes in $\mathbb{F}_2^N$, called small codes and big codes in [12].

The next lemma will help to further motivate our problem.

Lemma 3: Let A be the adjacency matrix of the graph $\mathsf{Cay}(\mathbb{F}_2^r, S)$ where S may or may not contain 0. If $n = |S|$ is odd then $\operatorname{rk} A = N = 2^r$ and if n is even then $AA^\intercal = 0$, implying in particular $\operatorname{rk} A \leq N/2$.

Proof: The rows and columns of A are indexed by vectors of $\mathbb{F}_2^r$. Two distinct rows x and y intersect in the set of positions $\mathcal{I} = (x+S) \cap (y+S)$ which is of even size because whenever $x + s_1 = y + s_2$ is in $\mathcal{I}$, so is $x+s_2 = y+s_1$, implying that $\mathcal{I}$ is partitioned into pairs of the form $\{z, z+(x+y)\}$. Therefore the matrix $AA^\intercal$ has only zeros outside the main diagonal. Now a diagonal element has value $\langle x, x\rangle = |S| \bmod 2$ so when n is odd we have $AA^\intercal = I_N$ and when n is even we have $AA^\intercal = 0$, hence the claims of the lemma. □

As a consequence, we observe that for a set of non-zero generators of *odd size* n, we have $\operatorname{rk} \tilde{A} \leq N/2$, whence $\dim\ker(\tilde{A}) \geq N/2$ so the rate of the storage code satisfies $R(\mathcal{C}) \geq 1/2$. It may happen that for some Cayley graphs of odd degree (not counting the loops) we have $\operatorname{rk} \tilde{A} < N/2$ in which case we will obtain an exceptional storage code of large rate. While this observation shows that this construction has a potential for uncovering large-size storage codes, it does not necessarily make finding such codes a straightforward task: in fact, as we have mentioned, for some years it was believed that they did not exist at all.

B. Coset Graphs of Binary Codes: The Quantum Coding Connection

The construction of quantum codes of Calderbank, Shor, and Steane [7], [29] relies on a pair of classical codes $\mathcal{C}_0, \mathcal{C}_1$ such that $\mathcal{C}_0^\perp \subset \mathcal{C}_1$, and it gives rise to a quantum code of dimension equal to $\dim(\mathcal{C}_1/\mathcal{C}_0^\perp)$ and distance $\min(w(\mathcal{C}_1 \backslash \mathcal{C}_0^\perp), w(\mathcal{C}_0 \backslash \mathcal{C}_1^\perp))$, where $w(\cdot)$ is the minimum Hamming weight of the argument set. Of interest to us is the particular case of $\mathcal{C}_0 = \mathcal{C}_1$. In other words, we start with a binary linear code $\mathcal{C} \subset \mathbb{F}_2^N$ generated by a matrix A such that $AA^\intercal = 0$, which ensures that $\mathcal{C}^\perp \subset \mathcal{C}$. The code $\mathcal{C}$ defines a quantum code through the above construction, and the associated quantum code has parameters $[[N, N - 2\operatorname{rk} A, D]]$, i.e. length N, dimension $N - 2\operatorname{rk} A$ and minimum distance D equal to the smallest Hamming weight of a vector in $\mathcal{C}^\perp \setminus \mathcal{C}$.

Therefore, for Cayley graphs over $\mathbb{F}_2^r$, the storage codes we are interested in also define quantum codes. From the point of view of storage, we do not have much use for the quantum code's minimum distance, but we are very much interested in its dimension: in particular we will obtain a storage code of rate greater than $1/2$ if and only if the associated quantum code has non-zero dimension. Quantum codes arising from Cayley graphs over $\mathbb{F}_2^r$ were studied in [12] and we will make use of some of results obtained in that work.

We now focus on the case of coset graphs of repetition codes of odd length $n = r+1$, since they form a natural generalization of the Clebsch graph which is the coset graph of the $[5, 1]$ repetition code. Alternatively, they can be seen as Cayley graphs $\mathsf{Cay}(\mathbb{F}_2^r, S)$, r even, with generating set $S = \{e_1, e_2, \ldots, e_r, e_1 + e_2 + \cdots + e_r\}$, where $e_1, \ldots, e_r$ denote the canonical generators. Now it turns out that in the quantum coding context, coset graphs G_m of repetition codes of *even* length m were studied in [12], because their adjacency matrices $A(G_m)$ are self-orthogonal and therefore directly yield quantum codes. These quantum codes were proved in [12] to have parameters $[[2^{m-1}, 2^{m/2}, 2^{m/2-1}]]$. We shall use this result to prove

Proposition 4: The storage codes associated to coset graphs of repetition codes of odd length $n \geq 5$ have length $N = 2^{n-1}$ and dimension $K = 2^{n-2} + 2^{(n-3)/2}$.

Proof: We start with the graphs G_m defined in the previous paragraph for even m. They are Cayley graphs with vertex set $\mathbb{F}_2^{m-1}$, generator set $S_m = \{e_1, \ldots, e_{m-1}, e_1 + \ldots + e_{m-1}\}$, and we note that all these generators are of odd weight since m is even. Therefore G_m is bipartite, and each of its edges connects an even-weight vertex to an odd-weight vertex. The adjacency matrix of G_m can therefore be written as

$$A(G_m) = \begin{bmatrix} 0 & B_m^\intercal \\ B_m & 0 \end{bmatrix}$$

where the rows of B_m are indexed by even-weight vectors of length $m-1$, its columns by odd-weight vectors, and B_m has a 1 in the coordinate indexed by row x and column y if and only if the vector $x+y \in S_m$. From the result of [12] quoted before the proposition we find that $\operatorname{rk} A(G_m) = \frac{1}{2}(2^{m-1} - 2^{m/2}) = 2^{m-2} - 2^{\frac{m}{2}-1}$.

We will now make a transition to odd length $n = m-1$ by transforming the odd-weight vector indices of the columns of B_m into even-weight vectors through the correspondence $x \mapsto \mathbf{1} + x$, where $\mathbf{1}$ denotes the all-one vector of length $n = m-1$. After permuting columns, the matrix B_m can now be seen as the adjacency matrix of the Cayley graph defined over the binary group of even-weight vectors of length $m-1$, with generators $\mathbf{1} + e_1, \mathbf{1} + e_2, \ldots, \mathbf{1} + e_n, 0$. Since n is odd we have

$$\sum_{i=1}^{n} (\mathbf{1} + e_i) = 0$$

and this last Cayley graph is therefore isomorphic to the Cayley graph over $\mathbb{F}_2^{n-1}$ with generators

$$0, e_1, \ldots, e_{n-1}, e_1 + \cdots + e_{n-1}.$$

The matrix B_m is therefore the parity-check matrix of the storage code $\mathcal{C}$ associated to the coset graph of the repetition code of *odd* length $n = m-1$ (note that this matrix has a unit diagonal). The length of the code $\mathcal{C}$ is $N = 2^{n-1}$, and the dimension

$$\begin{aligned} K &= \dim\ker(B_m) = N - \frac{1}{2}\operatorname{rk} A(G_m) \\ &= 2^{n-1} - \frac{1}{2}(2^{n-1} - 2^{\frac{n+1}{2}-1}) \\ &= 2^{n-2} + 2^{(n-3)/2} \end{aligned}$$

as was to be proved. □

This last proposition therefore yields an infinite family of graphs for which the associated storage codes have rate exceeding $1/2$. In particular, for $n = 5$ we recover the value of the rate $\frac{5}{8}$. As n increases, the rate

$$R(G_m) = \frac{1}{2} + \frac{1}{2^{(n+1)/2}}$$

decreases to 1/2, so the highest rate within this family is achieved for $n = 5$, i.e. for the Clebsch graph.

IV. Necessary Conditions for High Rate of Codes on Coset Graphs

We maintain the notation and conventions of Sec. III-A. Below we identify binary vectors $z \in \mathbb{F}_2^{2^r}$ with functions

$$f : \mathbb{F}_2^r \to \mathbb{F}_2,$$

namely we put $f(x) = z_x$, where $x \in \mathbb{F}_2^r$ is the (vector) index of the coordinates of z. Therefore the adjacency matrix A can be viewed as an operator on the function space $\mathcal{F}(\mathbb{F}_2^r) := \mathbb{F}_2^{\mathbb{F}_2^r}$ acting by

$$\begin{aligned} A\colon \mathcal{F}(\mathbb{F}_2^r) &\to \mathcal{F}(\mathbb{F}_2^r) \\ f &\mapsto Af \end{aligned}$$

with

$$Af(x) = \sum_{s\in S} f(x+s). \qquad (7)$$

Keeping the graph $G = \mathsf{Cay}(\mathbb{F}_2^r, S)$ in mind, we also call A the *adjacency operator* of G.

The starting observation in this part is given by the following proposition, due to Lowzow [21], as is point (a) of Theorem 6 below, that was formulated in a quantum coding context.

Proposition 5: Let V be a vector subspace of $\mathbb{F}_2^r$. If $|S \cap V|$ is odd, then $\operatorname{rk} A \geq 2^{\dim V}$.

Proof: Let us look at the effect of A on the restricted set of functions $F_V := \{f : \operatorname{supp}(f) \subset V\}$. We note that for every $x \in V$ we have

$$Af(x) = \sum_{s\in S\cap V} f(x+s)$$

since for $x \in V$ and $s \notin V$ we have $x+s \notin V$ and $f(x+s) = 0$ by our hypothesis on f. Therefore, when thus restricted to functions $V \to \mathbb{F}_2$, A acts on F_V as the adjacency operator $A_{S\cap V}$ of the Cayley graph defined by the set of generators $S\cap V$. Therefore the image of A has dimension at least equal to the dimension of the image of $A_{S\cap V}$. Now, Lemma 3 says that when $|S\cap V|$ is odd the image of $A_{S\cap V}$ has dimension $2^{\dim V}$, since $A_{S\cap V}$ is full-rank on the space of functions $V \to \mathbb{F}_2$. □

Definition 1: The *Schur product* of two codes $A, B \subset \mathbb{F}_2^n$ is a binary linear code $C = A * B \subset \mathbb{F}_2^n$ generated by all coordinatewise products $a * b = (a_1b_1, \dots, a_nb_n), a = (a_1, \dots, a_n) \in A, b = (b_1, \dots, b_n) \in B$.

Theorem 6: Let $G = \mathsf{Cay}(\mathbb{F}_2^r, S)$ be a Cayley graph over $\mathbb{F}_2^r$ with set of generators S containing the 0 element. Let H be the $r \times n$ matrix whose columns are made up of the non-zero elements of S, so that $n = |S| - 1$, and let C be a code of length n with parity-check matrix H. Finally, let $\mathcal{C}$ be the storage code on G. Then

(a) If $R(\mathcal{C}) > 1/2$, then n is odd and all the rows of H have even weight;

(b) If $R(\mathcal{C}) > (2^k - 1)/2^k, k = 2, 3, \dots$, then $(C^\perp)^{*(k-1)} \subset C$.

Proof: (a) Lemma 3 implies that $|S|$ is even and therefore n is odd. Let V_i be the subspace of $\mathbb{F}_2^r$ generated by all vectors of weight 1 except the vector with a 1 in coordinate i. We have $\dim V_i = r - 1$. Our assumption of $R(\mathcal{C}) > 1/2$ implies that $\operatorname{rk} A(G) < 2^{r-1}$, and then by Proposition 5, $|S \cap V_i|$ must be even, and so must therefore be $|S \cap \overline{V_i}|$. But $S \cap \overline{V_i}$ is exactly the set of columns of H that have a 1 in row i.

(b) We argue by induction. For the base case of $k = 2$ we need to show that $C^\perp \subset C$. Let i, j be distinct (row) indices, $1 \leq i, j \leq r$. Let V be the subspace of $\mathbb{F}_2^r$ generated by the basis vectors e_s except for e_i, e_j. As above, Proposition 5 implies that $|S \cap \overline{V}|$ is even. In other words, the set of columns of H having a 1 in rows h_i or h_j is even, i.e., $|h_i \cup h_j|$ is even. Since the weights of the rows h_i and h_j are also even by Part (a), this implies that $|h_i \cap h_j|$ is even, i.e., $\langle h_i, h_j\rangle = 0$. The row space of H is therefore included in its orthogonal space, proving the claim.

We will prove the induction step for $k = 3$ to ease notation. Take three distinct row indices i_1, i_2, i_3 and let V be generated by all the basis vectors except $e_{i_j}, j = 1, 2, 3$. As before, $|S \cap \overline{V}|$ is even, or in other words, $|\cup_{j=1}^3 h_{i_j}|$ is even. Now $|h_{i_1} \cap h_{i_2}|$ and $|h_{i_1} \cap h_{i_3}|$ are even by the base case, and therefore $|h_{i_1} \cap h_{i_2} \cap h_{i_3}|$ is also even, i.e., $(h_{i_1} * h_{i_2}, h_{i_3}) = 0$. This shows that $C^\perp$ is orthogonal to $C^\perp * C^\perp$, meaning that $C^\perp * C^\perp \subset (C^\perp)^\perp = C$. For general k we just use the fact that the l-wise intersections of the rows are even for all $1 \leq l \leq k - 1$, and thus so is the k-wise intersection. □

V. A Family of Storage Codes on Triangle-Free Graphs of Rate 3/4

In this section we consider a family of storage codes on coset graphs of a specially constructed family of binary codes C_r of length $n = 2^{r-1} + 1$, dimension $k = 2^{r-1} - r$, and distance 4. To define it, start with the parity-check matrix of the extended Hamming code of length 2^{r-1}, augment it with an all-zero column, and then add a row of weight 2 that contains a '1' in the last position. Denote the resulting $(r + 1) \times n$ matrix by H_r. For instance, for $r = 4$ we obtain the matrix

$$H_4 = \left[\begin{array}{cccccccc|c} 0&0&0&0&1&1&1&1&0\\ 0&0&1&1&0&0&1&1&0\\ 0&1&0&1&0&1&0&1&0\\ 1&1&1&1&1&1&1&1&0\\ \hline 0&0&0&0&0&0&0&1&1 \end{array}\right] \qquad (8)$$

where the choice of the last row is largely arbitrary as long as it is of weight 2 and intersects the all-zero column.

Observe that the code C_r does not contain its dual $C_r^\perp$, i.e., the code generated by H_r, because for instance the last row of H_r is not orthogonal to the other rows. Thus, from Theorem 6, the most we can hope for the storage code constructed on the coset graph of C_r is rate $3/4$. In this section we prove that the rate $R(\mathcal{C}_r)$ is in fact close to this maximum value.

Theorem 7: Let $\mathcal{C}_r$ be the $[N = 2^{r+1}, K]$ storage code constructed on the coset graph of the code $C_r, r \geq 4$. Then

$$\frac{K}{N} = \frac{3}{4} - \frac{1}{2^r}.$$

The idea behind the construction is that the kernel of the adjacency operator associated to the coset graph of the extended Hamming code is easily described, and that the matrix (8) is essentially obtained by adding an extra row to the parity-check matrix of the extended Hamming code: we can therefore hope to compute the dimension of the kernel of the adjacency operator associated to this new code from the adjacency operator for the coset graph of the extended Hamming code. Now the latter graph is simply a complete bipartite graph whose adjacency matrix is

$$A = \begin{bmatrix} 0 & J \\ J & 0 \end{bmatrix}$$

where J is the all-one matrix of order 2^{r-1}. This is evident once we realize that the elements of $\mathbb{F}_2^r$ of the form $(*, \dots, *, 0)$ are connected to all the elements $(*, \dots, *, 1)$, while at the same time having no edges among themselves. The kernel of A is formed of vectors of length 2^r of even weight in the first half as well as in the second half, and therefore has dimension $2^r - 2$. To switch from the coset graph of the extended Hamming code to the coset graph of the code C_r we need to understand the effect on the adjacency operators of adding an extra coordinate to a set of generators.

Lemma 8 that we derive below will go some way towards giving a formula for deriving the new dimension of the kernel of the adjacency operator when an extra coordinate is added to the set of generators. To introduce it we need some additional notation.

Let $\tilde{S}$ be any subset of $\mathbb{F}_2^{r+1}$, and let $\tilde{A}$ be the adjacency operator (7) on the space of functions $\mathbb{F}_2^{r+1} \to \mathbb{F}_2$, associated to the Cayley graph $\mathsf{Cay}(\mathbb{F}_2^{r+1}, \tilde{S})$. Let $\pi\colon \mathbb{F}_2^{r+1} \to \mathbb{F}_2^r$ be the projection on the first r coordinates, i.e. $\pi(x)$ is obtained from the vector x by removing its last coordinate. Now let $S = \pi(\tilde{S})$. Some caution is in order here, because it may happen that two distinct elements of $\tilde{S}$ project onto the same vector if they differ only in their last coordinate. We consider S as a multiset, i.e. we do not collapse into one element the images by π of two distinct elements if they happen to have the same projection, therefore we have in particular $|S| = |\tilde{S}|$. Now it may make little sense to consider the Cayley graph over $\mathbb{F}_2^r$ with a generator set S that contains duplicate elements, but we can nevertheless define the operator A on the space of functions $\mathbb{F}_2^r \to \mathbb{F}_2$, associated to S through the formula (7). We note in passing that A is equal to the adjacency operator of the Cayley graph over $\mathbb{F}_2^r$ with the generator set obtained from S by removing duplicate elements.

Next, for $i = 0, 1$, let $\tilde{S}_i$ denote the subset consisting of the elements of $\tilde{S}$ with a i in the last coordinate, and let $S_0 = \pi(\tilde{S_0})$ and $S_1 = \pi(\tilde{S_1})$. Let A_0 and A_1 be the associated adjacency operators, i.e. the adjacency operators of the Cayley graphs $\mathsf{Cay}(\mathbb{F}_2^r, S_0)$ and $\mathsf{Cay}(\mathbb{F}_2^r, S_1)$. The operators A_0 and A_1 act therefore on the space of functions $\mathbb{F}_2^r \to \mathbb{F}_2$. Note that we have the decomposition:

$$A = A_0 + A_1.$$

Let e denote the vector of weight 1 in $\mathbb{F}_2^{r+1}$ supported by the last coordinate, and let V denote the subspace of $\mathbb{F}_2^{r+1}$ supported by the r old coordinates, isomorphic therefore to $\mathbb{F}_2^r$. Recall that $\mathcal{F}(\mathbb{F}_2^r)$ denotes the space of functions $\mathbb{F}_2^{\mathbb{F}_2^r}$. Since $\mathcal{F}(\mathbb{F}_2^{r+1})$ is formed of two copies of $\mathcal{F}(\mathbb{F}_2^r)$, we have the isomorphism

$$\begin{aligned} \mathcal{F}(V) \times \mathcal{F}(V) &\xrightarrow{\sim} \mathcal{F}(\mathbb{F}_2^{r+1}) \\ (\mathbf{1}_X, \mathbf{1}_Y) &\mapsto \mathbf{1}_X + \mathbf{1}_{Y+e} \end{aligned} \tag{9}$$

where X is the support of the first half of the function (vector) and Y the support of its second half. Now we may identify any function $\tilde{f} \in \mathcal{F}(\mathbb{F}_2^{r+1})$ with a pair of functions (f_0, f_1) through the above isomorphism. Spelling it out:

$$\begin{aligned} f_0\colon V &\to \mathbb{F}_2 \\ x &\mapsto f_0(x) = \tilde{f}(x) \\ f_1\colon V &\to \mathbb{F}_2 \\ x &\mapsto f_1(x) = \tilde{f}(x+e) \end{aligned}$$

The matrix $\tilde{A}$ has the form $\tilde{A} = \begin{bmatrix} A_0 | A_1 \\ \hline A_1 | A_0 \end{bmatrix}$, and so we have:

$$(\tilde{A}\tilde{f})_0 = A_0 f_0 + A_1 f_1 \tag{10}$$

$$(\tilde{A}\tilde{f})_1 = A_0 f_1 + A_1 f_0 \tag{11}$$

We can now derive:

Lemma 8: The adjacency operator $\tilde{A}$ associated to the set of generators $\tilde{S}$ satisfies

$$\begin{aligned} \dim\ker\tilde{A} = \dim\ker A &+ \dim(\ker A_1 \cap \ker A) \\ &+ \dim(\operatorname{Im} A \cap \operatorname{Im} A_{0|\ker A}), \end{aligned} \tag{12}$$

where $A_{0|\ker A}$ is the restriction of A_0 to the kernel of A.

Proof: Let F_D (D for double) be the set of functions in $\mathcal{F}(V) \times \mathcal{F}(V)$ of the form (f, f), and let F_L (L for left) be the set of functions of the form $(f, 0)$ and note that $F_D \cap F_L = \{0\}$. Any function $\tilde{f} \in \mathcal{F}(\mathbb{F}_2^{r+1})$, identified with $\mathcal{F}(V) \times \mathcal{F}(V)$ through the isomorphism (9), has a unique decomposition as $\tilde{f} = \tilde{f}_D + \tilde{f}_L$, with $\tilde{f}_D \in F_D$ and $\tilde{f}_L \in F_L$. Therefore, $\tilde{f} \in \ker\tilde{A}$ if and only if $\tilde{A}\tilde{f}_D = \tilde{A}\tilde{f}_L$. We therefore have

$$\begin{aligned} \dim\ker\tilde{A} = \dim\ker\tilde{A}_{|F_D} &+ \dim\ker\tilde{A}_{|F_L} \\ &+ \dim(\operatorname{Im}\tilde{A}_{|F_D} \cap \operatorname{Im}\tilde{A}_{|F_L}). \end{aligned}$$

On account of (10) and (11) any function $\tilde{f} \in F_D$ is in $\ker\tilde{A}$ iff $(A_0 + A_1) f_0 = A f_0 = 0$, i.e. $f_0 \in \ker A$. Therefore

$$\dim\ker\tilde{A}_{F_D} = \dim\ker A.$$

Next, we consider functions $\tilde{f}$ for which $f_1 = 0$. Again by (10) and (11) a function $\tilde{f} \in F_L$ is in $\ker\tilde{A}$ iff $f_0 \in (\ker A_0 \cap \ker A_1)$. From $A = A_0 + A_1$ we have that $\ker A_0 \cap \ker A_1 = \ker A_1 \cap \ker A$, and so

$$\dim\ker\tilde{A}_{F_L} = \dim(\ker A_1 \cap \ker A).$$

It remains to compute the dimension of $\operatorname{Im}\tilde{A}_{|F_D} \cap \operatorname{Im}\tilde{A}_{|F_L}$. Let $\tilde{f} = (f, f) \in F_D$, then from (10) and (11) we have $\tilde{A}\tilde{f} \cong (Af, Af)$.

Now let $\tilde{f}' = (f', 0) \in F_L$, then, again from (10)-(11), $\tilde{A}\tilde{f}' \cong (A_0 f', A_1 f')$. So $\tilde{A}\tilde{f}' \in \operatorname{Im} F_D$ if and only if $A_0 f' \in \operatorname{Im} A$ and $A_0 f' = A_1 f'$. This last condition is equivalent, since $A_0 + A_1 = A$, to $f' \in \ker A$. Therefore, $\operatorname{Im} F_D \cap \operatorname{Im} F_L \simeq \operatorname{Im} A \cap \operatorname{Im} A_{0|\ker A}$. □

Proof of Theorem 7: Let $\tilde{S} \subset \mathbb{F}_2^{r+1}$ consist of the columns of the parity-check matrix H_r of C_r to which we add the zero vector, so that the storage code associated to C_r is the kernel of the operator $\tilde{A}$ associated to $\tilde{S}$. We now apply Lemma 8 to $\tilde{A}$ and compute the dimensions of the spaces on the right-hand side of (12).

Removing the last coordinate to the elements of $\tilde{S}$ yields a set S equal to the set of columns of the parity-check matrix of the extended Hamming code, plus two copies of the zero vector. The associated operator A is therefore equal to the adjacency operator of the Cayley graph over $\mathbb{F}_2^r$, with generator set equal to all vectors of $\mathbb{F}_2^r$ that end with a 1 on their last coordinate. As noted earlier, this graph is the complete bipartite graph on $\mathbb{F}_2^{r-1}$, and the kernel of A is the space of functions that are of even weight on each of the spaces V_0 and V_1, corresponding to the set of vectors $(x_1, \dots, x_r)$ satisfying $x_r = 0$ and $x_r = 1$, and its dimension equals

$$\dim \ker A = 2^r - 2.$$

The last row of the matrix H_r is of weight 2, so there are exactly two elements of $\tilde{S}$ that end with a 1 in their last coordinate, which means that $|S_1| = 2$. Furthermore, one of these two elements of $\tilde{S}$ projects onto the zero vector of $\mathbb{F}_2^r$ by π, so that $S_1 = \{0, s\}$ for some non-zero vector $s \in \mathbb{F}_2^r$, e.g., in (8) $s = \mathbf{1}^r$. Applying (7) we have

$$A_1 f(x) = f(x) + f(x+s).$$

We therefore have that $\ker A_1$ is equal to the space of functions whose support in $\mathbb{F}_2^r$ is stable by addition by s. Now, since s by construction must be equal to a column of the parity-check matrix of the extended Hamming code, it ends with a 1, i.e. is of the form $s = (s_1, \dots, s_{r-1}, s_r = 1)$. Therefore, a function is in $\ker A_1$ if and only if its support is equal to $X \cup (X+s)$ for $X \subset V_0$ and $X + s \subset V_1$. Intersecting $\ker A_1$ with $\ker A$ restricts the sets X to sets of even size, and thus

$$\dim(\ker A_1 \cap \ker A) = \dim \ker A_1 - 1 = 2^{r-1} - 1.$$

For the third term in (12) observe that $A_1 \mathbf{1}_{V_0} = \mathbf{1}_{V_0} + \mathbf{1}_{V_1} = \mathbf{1}_V \in \operatorname{Im} A \cap \operatorname{Im} A_{1|\ker A}$. Furthermore, we clearly have that $\operatorname{Im} A_1$ must consist of functions whose support is stable by addition by s, i.e. $\operatorname{Im} A_1 = \ker A_1$, implying that $\mathbf{1}_{V_0}$ is in $\operatorname{Im} A$ but not in $\operatorname{Im} A_1$. Since $\dim(\operatorname{Im} A) = 2$, we have therefore that $\operatorname{Im} A \cap \operatorname{Im} A_{1|\ker A}$ is equal to the space of constant functions and of dimension 1. Since $A_0 + A_1 = A$, we have that A_0 and A_1 are equal on $\ker A$, and

$$\dim(\operatorname{Im} A \cap \operatorname{Im} A_{0|\ker A}) = 1.$$

Collecting the terms in (12), we obtain

$$\dim \ker \tilde{A} = 2^r + 2^{r-1} - 2,$$

which yields the rate expression in the theorem. □

VI. Correcting Multiple Erasures

A. The Edge-Vertex Construction and Graph Expansion

The most likely failure scenario in storage applications is loss of a single node, which corresponds to correcting a single erasure in the codeword of a storage code. This question is central to the paper that defined codes with local erasure correction [16], and it was also one of the original motivations for studying such codes on graphs in the paper by Mazumdar [23]. At the same time, correcting multiple erasures represents a natural extension of this problem, which was addressed in a large number of papers on codes with locality, among them [20], [30]. Motivated by this research, [23] posed the question of recovering the codeword when multiple vertices are erased. This question can be posed in several ways based on the vertex recovery procedure, and we proceed to specify our assumptions. Throughout this section we limit ourselves to d-regular graphs and to codes obtained from the edge-vertex construction of Section II-B.

We begin with a code $\mathcal{C}(G) = \{x, x \in \mathbb{F}_q^{d\cdot|V|}\}$ defined on a graph $G(V,E)$, where as before, the coordinates of the codeword are placed on the vertices of G. The coordinates x_v can be viewed either as d-vectors over $\mathbb{F}_q$ or as elements of $\mathbb{F}_{q^d}$, and they are obtained by collecting all the values placed on the edges incident to v. Extending the definition of storage codes, suppose that every vertex $v \in V$ can recover its value x_v from the values of its neighbors $x_u, u \in \mathcal{N}(v)$ as long as at most t of them are unavailable. Of course, taking $t = 0$ takes us back to the original definition of storage codes. The easiest form of the multiple erasure correction problem arises if we assume that every erased vertex is adjacent to at most t other erased vertices. To address it, we simply place constraints on the edges in the neighborhood of every vertex: namely, assume that the symbols placed on the edges in $E(v)$ form a vector in a linear code D of length d over $\mathbb{F}_q$ that corrects t erasures. This defines a linear storage code $\mathcal{C} = \mathcal{C}(G, D)$ that enables each vertex to recover its value from $d - t$ or more nonerased neighbors. The dimension of the code $\mathcal{C}$ is at least $|V|(2R(D) - 1)$, where $R(D) := \frac{\dim(D)}{d}$ is the rate of the local code D.

This approach allows correction only of erasure patterns that leave $d - t$ or more neighbors of each vertex nonerased, which is a somewhat artificial assumption (indeed, the erased vertices may not follow the connections in the graph). Lifting it, we will assume next that the set of erased vertices $S \subset V$ is of arbitrary shape, and engage standard ideas from error-correcting codes on graphs, in particular, graph expansion and its use in decoding [28]. Namely, instead of specifying that the vertices can correct themselves independently, we will be satisfied if there is one erased vertex with t or fewer neighbors in the subgraph induced by S. Once it is recovered, there are fewer erased vertices (edges), and (under some conditions) there will be more erased vertices that can correct themselves from their neighborhoods.

As remarked, this approach is close to the classic error-correcting codes on graphs, also known as the Tanner codes [31]. The only difference between them and our construction of storage codes is related to the way erasures are applied to the codeword coordinates. Namely, while in earlier works the edges were erased independently of each other based on the properties of the communication channel, in storage codes erasures affect the vertices, which results in erasing all the values of the edges incident to the vertex at once.

The edge-vertex construction enables us to iteratively correct multiple erasures if we assume that the underlying graph has expansion properties. We will use the following well-known result.

Lemma 9: (Expander Mixing Lemma [19, Lemma 2.5]) Let G be a d-regular graph with N vertices and eigenvalues $\lambda_1 = d \geq \lambda_2 \geq \cdots \geq \lambda_N$. Then for any $U, T \subset V$

$$\Big| |E(U,T)| - \frac{d|U||T|}{N} \Big| \leq \lambda\sqrt{|U||T|},$$

where $\lambda := \max(|\lambda_2|, |\lambda_N|)$.

Proposition 10: Consider a storage code $\mathcal{C}(G,D)$ defined by the edge-vertex construction. Suppose that G is a d-regular graph with N vertices and the local code D corrects t erasures. Let $U \subset V, |U| = \sigma N$ be the set of erased vertices. As long as

$$\sigma \leq \frac{t}{d} - \frac{\lambda}{d}, \tag{13}$$

the erased vertices can be recovered based on the local iterative procedure.

Proof: Taking $T = U$ in Lemma 9, we obtain for the number of edges in the subgraph induced by U the inequality

$$2|E(U)| \leq \frac{d}{N}|U|^2 + \lambda|U|.$$

Let $\partial(U) = \{(u,v) \in E(G) : u \in U, v \in V \backslash U\}$ be the *edge boundary* of U. We have

$$d|U| = 2|E(U)| + |\partial(U)|.$$

Taken together, this implies

$$\frac{|\partial(U)|}{\sigma N} \geq d(1-\sigma) - \lambda.$$

In other words, there exists a vertex $v \in U$ with at least $d(1-\sigma) - \lambda$ nonerased neighbors. As long as

$$d(1-\sigma) - \lambda \geq d - t,$$

its value can be recovered using the local code D. Under the assumption (13) this inequality is satisfied, so size of the erased set of vertices has decreased. The remaining set of erased vertices $U' = U \backslash \{v\}$ also satisfies (13), and the proof is concluded by straightforward induction. □

To use this result, we need the spectral gap $d - \lambda$ to be large compared to d or λ much smaller than d. The limits for the spectral gap are given by the Alon-Boppana bound [19, Thm.2.7], namely for any d-regular graph on n vertices

$$\lambda \geq 2\sqrt{d-1} - o_n(1).$$

The graphs with $\lambda \leq 2\sqrt{d-1}$ are known to exist by the Lubotzky-Phillips-Sarnak and Margulis constructions [19, Sec.5.11]. Assuming that the graph G is from this family, we can state the following:

Corollary 11: There exist storage codes $\mathcal{C}(G,D)$ on d-regular graphs with local codes correcting t erasures that recover from any pattern of s erased vertices as long as their proportion $\sigma = s/n$ satisfies

$$\sigma \leq \frac{t}{d} - O(d^{-1/2}).$$

Note that large spectral gap guarantees that a subset U has many edges that connect it with its complement, or in other words, its *edge expansion ratio* is large. Further connections between the spectral gap and expansion are discussed in [19, Sec. 4.5].

VII. Discussion

A. Numerical Experiments

It is tempting to look for other coset graphs of linear codes whose full-parity storage code has rate greater than $1/2$. It is a challenge to derive general formulae for dimensions however. With computer help we find:

Proposition 12:

(a) The binary Golay code of length 23 and dimension 11 yields a graph G on $N = 2048$ vertices that supports a linear storage code of rate $41/64$.

(b) The 2-error-correcting binary BCH code of length $n = 2^s - 1$ and dimension $k = 2^s - 1 - 2s$ yields a graph G_s with $N = 2^{2s}$ vertices that supports a linear storage code with rate given in the following table

s	4	5	6	7	8
$R_2(G_s)$	$\frac{39}{64}$	$\frac{347}{512}$	$\frac{1497}{2048}$	$\frac{6387}{8192}$	$\frac{26859}{32768}$.

The sequence of rate values obtained in Proposition 12 is $0.6094, 0.6777, 0.7309, 0.7796, 0.8196$, and the value $R_2(G_8) = 0.8196$ of the storage code of length $N = 65536$ now represents the largest known rate of storage codes on triangle-free graphs over any alphabet.

B. Open Problems

1. *The rate question:* Arguably, the central question is whether rates of storage codes on triangle-free graphs can be arbitrarily close to 1. If not, what is an upper limit for these rates?

2. *BCH codes:* In view of the numerical experiments it is of interest to find or estimate the dimension of storage codes obtained from the 2-error-correcting BCH codes. More specifically, what is $\limsup_{s\to\infty} R_2(G_s)$? At this point we cannot even rule out that the sequence $(R_2(G_s))_s$ in the limit reaches 1, which would resolve the question of the maximum possible rate of storage codes on triangle-free graphs.

3. *Reed-Muller codes:* Another good candidate arises from the family of Reed-Muller codes. Fix $m \geq 4$ and consider Boolean polynomials $f(v) : \mathbb{F}_2^m \to \mathbb{F}_2$. The second order Reed-Muller code $RM(m,2)$ is spanned by the set of evaluations of functions of degree ≤ 2 on all the elements of $\mathbb{F}_2^m$. Consider a subcode C_m of the punctured code $RM(m,2)$ spanned by the evaluations of the functions $v_i, 1 \leq i \leq m$ and $v_iv_j, 1 \leq i < j \leq m$ on the nonzero points of $\mathbb{F}_2^m$. The length of the code C_m is $2^m - 1$ and its dimension is $m(m+1)/2$. Another way of constructing this code is to take a linear space spanned by the Simplex code S_m and its Schur square $S_m * S_m$.

Now consider the code $(C_m)^\perp$, i.e., a code whose parity check matrix H has rows given by the evaluations of the linear and quadratic functions. This is a code of odd length with even-weight parities, and it also contains its dual as well as Schur powers of the dual, fulfilling the necessary conditions for high-rate storage codes in Theorem 6. Finding the actual rate of the storage code $\mathcal{C}$ is however not straightforward, and we leave this as an open problem.

C. Erasure Correction and Bootstrap Percolation

We conclude with one more observation regarding correcting multiple erasures with storage codes. Given a graph G, we again assume the edge-vertex construction that relies on a local code correcting t erasures. Assume that the vertices are erased randomly and independently with some probability $\bar{p}$, forming a set $U \subset V$ of erased vertices. We wish for the iterative procedure employed in the previous section to successively recover the erased vertices until all are corrected. Rephrasing, suppose that functional vertices are selected randomly with probability $p = 1 - \bar{p}$, and an erased vertex with not more than t erased neighbors recovers its value, becoming functional. In graph-theoretic terms this procedure is known as *bootstrap percolation,* and it represents an established branch of percolation theory. Percolation occurs when all the erased vertices have been recovered (for infinite graphs, recovered with probability one). This problem was introduced in [9]; see the recent paper [17] for an overview of the main results. The main problem studied in the literature is the determination of the *critical probability*, defined as

$$p_c := \inf\{p : \mathbb{P}_p(U \text{ is corrected by the iterative process}) \ge 1/2\}.$$

The known results include the determination of p_c for the infinite grid $[n]^d$ as $n \to \infty$ (see [18] for $d = 2$ and [4] for all d) as well as for several other classes of graphs. These results translate directly to the corresponding thresholds for erasure correction with storage codes on graphs in which the vertices are erased randomly and independently.

Bootstrap percolation has been also considered in the deterministic setting [11], [13], where the main question is finding the minimum size of a set of functional vertices that enables the code to correct all erasures, or, using the language of the cited papers, the minimum number of *infected vertices* that infect the entire graph. Rephrasing, this means that we attempt to pinpoint a particular combination of erasures of the largest possible size that can be recovered through the iterative t-neighbor process. This problem, however, is an opposite of the natural question addressed by storage codes, where one is interested in the largest s such that *any* combination of s erased vertices can be recovered, or the smallest size of the set of nonerased vertices that enable recovery of the entire graph irrespective of the shape of the erased set U.

References

[1] R. Gummadi, A. Shokrollahi, and R. Sreenivas, "Broadcasting with side information," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Jan. 2010, pp. 823–832.

[2] F. Arbabjolfaei and Y.-H. Kim, "Three stories on a two-sided coin: Index coding, locally recoverable distributed storage, and guessing games on graphs," in *Proc. 53rd Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sep. 2015, pp. 843–850.

[3] F. Arbabjolfaei and Y.-H. Kim, "Fundamentals of index coding," *Found. Trends Commun. Inf. Theory*, vol. 14, nos. 3–4, pp. 163–346, 2018.

[4] J. Balogh, B. Bollobás, H. Duminil-Copin, and R. Morris, "The sharp threshold for bootstrap percolation in all dimensions," *Trans. Amer. Math. Soc.*, vol. 364, no. 5, pp. 2667–2701, May 2012.

[5] A. Blasiak, R. Kleinberg, and E. Lubetzky, "Broadcasting with side information: Bounding and approximating the broadcast rate," *IEEE Trans. Inf. Theory*, vol. 59, no. 9, pp. 5811–5823, Sep. 2013.

[6] A. Brouwer and W. Haemers, *Spectra of Graphs*. New York, NY, USA: Springer, 2012.

[7] A. R. Calderbank and P. W. Shor, "Good quantum error-correcting codes exist," *Phys. Rev. A, Gen. Phys.*, vol. 54, no. 2, pp. 1098–1105, Aug. 1996.

[8] P. J. Cameron, A. N. Dang, and S. Riis, "Guessing games on triangle-free graphs," *Electron. J. Combinatorics*, vol. 23, no. 1, pp. 1–9, Mar. 2016.

[9] J. Chalupa, P. L. Leath, and G. R. Reich, "Bootstrap percolation on a Bethe lattice," *J. Phys. C, Solid State Phys.*, vol. 12, no. 1, pp. L31–L35, Jan. 1979.

[10] D. Christofides and K. Markström, "The guessing number of undirected graphs," *Electron. J. Combinatorics*, vol. 18, no. 1, p. P192, Sep. 2011.

[11] A. Coja-Oghlan, U. Feige, M. Krivelevich, and D. Reichman, "Contagious sets in expanders," in *Proc. 26th Annu. ACM-SIAM Symp. Discrete Algorithms*, Philadelphia, PA, USA, 2015, pp. 1953–1987.

[12] A. Couvreur, N. Delfosse, and G. Zemor, "A construction of quantum LDPC codes from Cayley graphs," *IEEE Trans. Inf. Theory*, vol. 59, no. 9, pp. 6087–6098, Sep. 2013.

[13] D. Freund, M. Poloczek, and D. Reichman, "Contagious sets in dense graphs," *Eur. J. Combinatorics*, vol. 68, pp. 66–78, Feb. 2018.

[14] M. Gadouleau and S. Riis, "Graph-theoretical constructions for graph entropy and network coding based communications," *IEEE Trans. Inf. Theory*, vol. 57, no. 10, pp. 6703–6717, Oct. 2011.

[15] C. Godsil and G. Royle, *Algebraic Graph Theory*. New York, NY, USA: Springer-Verlag, 2001.

[16] P. Gopalan, C. Huang, H. Simitci, and S. Yekhanin, "On the locality of codeword symbols," *IEEE Trans. Inf. Theory*, vol. 58, no. 11, pp. 6925–6934, Nov. 2011.

[17] J. Gravner, C. Hoffman, J. Pfeiffer, and D. Sivakoff, "Bootstrap percolation on the Hamming torus," *Ann. Appl. Probab.*, vol. 25, no. 1, pp. 287–323, Feb. 2015.

[18] A. E. Holroyd, "Sharp metastability threshold for two-dimensional bootstrap percolation," *Probab. Theory Rel. Fields*, vol. 125, no. 2, pp. 195–224, Feb. 2003.

[19] S. Hoory, N. Linial, and A. Wigderson, "Expander graphs and their applications," *Bull. Amer. Math. Soc.*, vol. 43, pp. 439–561, Aug. 2006.

[20] G. M. Kamath, N. Prakash, V. Lalitha, and P. V. Kumar, "Codes with local regeneration and erasure correction," *IEEE Trans. Inf. Theory*, vol. 60, no. 8, pp. 4637–4660, Aug. 2014.

[21] C. L. Lowzow, "On quantum CSS-codes from Cayley graphs," M.S. thesis, Dept. Math., Université de Bordeaux, Bordeaux, France, 2020.

[22] A. Mazumdar, "On a duality between recoverable distributed storage and index coding," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun. 2014, pp. 1977–1981.

[23] A. Mazumdar, "Storage capacity of repairable networks," *IEEE Trans. Inf. Theory*, vol. 61, no. 11, pp. 5810–5821, Nov. 2015.

[24] A. Mazumdar, A. Mcgregor, and S. Vorotnikova, "Storage capacity as an information-theoretic vertex cover and the index coding rate," *IEEE Trans. Inf. Theory*, vol. 65, no. 9, pp. 5580–5591, Sep. 2019.

[25] S. Riis, "Information flows, graphs and their guessing numbers," *Electron. J. Combinatorics*, vol. 14, no. 1, p. 17, Jun. 2007.

[26] M. Rosenfeld, "Independent sets in regular graphs," *Isr. J. Math.*, vol. 2, no. 4, pp. 262–272, Dec. 1964.

[27] K. Shanmugam and A. G. Dimakis, "Bounding multiple unicasts through index coding and locally repairable codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun. 2014, pp. 296–300.

[28] M. Sipser and D. A. Spielman, "Expander codes," *IEEE Trans. Inf. Theory*, vol. 42, no. 6, pp. 1710–1722, Nov. 1996.

[29] A. Steane, "Multiple-particle interference and quantum error correction," *Proc. R. Soc. Lond. A, Math., Phys. Eng. Sci.*, vol. 452, pp. 2551–2577, Nov. 1996.

[30] I. Tamo and A. Barg, "A family of optimal locally recoverable codes," *IEEE Trans. Inf. Theory*, vol. 60, no. 8, pp. 4661–4676, Aug. 2014.

[31] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inf. Theory*, vol. IT-27, no. 5, pp. 533–547, Sep. 1981.