STORAGE CODES AND RECOVERABLE SYSTEMS ON LINES AND GRIDS

ALEXANDER BARG, OHAD ELISHCO, RYAN GABRYS, GEYANG WANG, AND EITAN YAAKOBI

ABSTRACT. A storage code is an assignment of symbols 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, or more generally, of a certain neighborhood of the vertex in G. In this work we introduce a new construction method of storage codes, enabling one to construct new codes from known ones via an interleaving procedure driven by resolvable designs. We also study storage codes on $\mathbb{Z}$ and $\mathbb{Z}^2$ (lines and grids), finding closed-form expressions for the capacity of several one and two-dimensional systems depending on their recovery set, using connections between storage codes, graphs, anticodes, and difference-avoiding sets.

1. INTRODUCTION

The concept of locality in coding theory has been the subject of extensive research during the last decade. Codes with local recovery were initially motivated by applications in distributed storage systems [17], wherein the lost data is recovered by accessing a small subset of the coordinates of the codeword, saving the volume of communication in the system aimed at performing the data reconstruction. Following their introduction, this class of codes was the subject of a large number of works which considered it from algebraic, combinatorial, and information-theoretic perspectives. We refer to a recent comprehensive monograph [24] for an overview of results and problems concerning these codes.

It was further observed in [21] that distributed storage systems may introduce additional constraints on communication performed for data recovery. The topology of the storage network may suggest that the nodes of the system be treated differently depending on their relative location, physical proximity, or network connectivity limitations. A natural way to model these limitations is to assume that the coordinates of the codeword correspond to the vertices of a graph that represents the connections in the system, and that the recovery of a coordinate utilizes the information available to it from its neighbors in the graph.

The main problem concerning storage codes is establishing the largest size of the code that supports local recovery for a given class of graphs. It was soon realized that this problem can be equivalently phrased as finding the smallest rate of symmetric index codes [4], or the largest success probability in (one variant of) guessing games on the graph G [26]; see [2, 7] for a more detailed discussion. Some high-rate storage codes were recently constructed in [6, 7, 16].

Alexander Barg is with ISR/Dept. of ECE, University of Maryland, College Park, MD, USA. Email: abarg@umd.edu.

Ohad Elishco is with Ben-Gurion University of the Negev, Israel, Email: ohadeli@bgu.ac.il.

Ryan Gabrys is with the University of California at San Diego, CA, USA. Email: ryan.gabrys@gmail.com.

Geyang Wang is with ISR/Dept. of ECE, University of Maryland, College Park, MD, USA. Email: wanggy@umd.edu.

Eitan Yaakobi is with the Technion - Israel Institute of Technology, Haifa, Israel. Email: yaakobi@cs.technion.ac.il.

The work of Alexander Barg was supported in part by NSF Grants CCF2104489, CCF2110113 (NSF-BSF), and CCF2330909. The work of Ohad Elishco was supported in part by NSF-BSF Grant CCF2020762. The work of Ryan Gabrys and Eitan Yaakobi was supported in part by NSF Grant CCF2212437. The work of Geyang Wang was supported in part by NSF-BSF Grant CCF2110113.

To facilitate the study of large-scale storage systems, two of the present authors suggested to extend the concept of codes with local recovery to infinite graphs, calling them *recoverable systems* [12]. The simplest model in this class arises when the vertices of the graph are taken to be all integer numbers $\mathbb{Z}$. To define data encoding on $\mathbb{Z}$, let us first introduce some notation. Let $\mathcal{Q}$ be a finite set (the code alphabet) and let $R \subset \mathbb{Z}$ be a finite set. Let $n+R := \{n+r \mid r \in R\}$. For a sequence $x \in \mathcal{Q}^V$ and for a set of integers R, denote by x_R the restriction of x to the positions in R. A recoverable system X is formed of bi-infinite sequences $x \in \mathcal{Q}^{\mathbb{Z}}$ with the property that for any $i \in \mathbb{Z}$ the value x_i is found as $f(x_{i+R})$, $R = \{j : 0 < |j| < l\}$ $(l \geq 1)$ and $f : \Sigma^{2l} \to \mathcal{Q}$ is a deterministic function, independent of i. They additionally assumed that the system X is shift invariant, i.e., if $x \in X$ then also a left shift $Tx \in X$ (T acts by shifting all the symbols in x one place to the left). This assumption enabled them to rely on methods from constrained systems to estimate the growth rate of the set of allowable sequences, or the capacity of recoverable systems.

The object of this paper is storage capacity of finite graphs, constructions of storage codes, as well as capacity and constructions of recoverable systems. For finite graphs we present a construction of codes based on interleaving known codes controlled by resolvable designs to obtain new codes from known ones. If the seed codes are optimal, then so are the interleaved ones. For graphs with transitive automorphisms, we phrase the capacity problem in terms of the code-anticode bound [11, 1].

For the infinite case, using finite subgraphs of $\mathbb{Z}$ and $\mathbb{Z}^2$, we obtain capacity values for certain recovery regions R. For instance, for $\mathbb{Z}_2$, we find storage capacity for balls in the l_1 and l_∞ metrics as well as certain cross-shaped regions. These results do not involve the shift invariance assumption.

Some of the results of this paper were presented at the 2022 IEEE International Symposium on Information Theory and published as an extended abstract [5]. Here we add new results, notably Section 3, and also provide complete or corrected proofs of the results announced in [5].

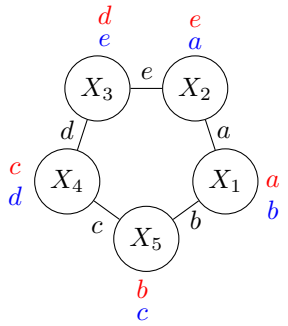
2. Preliminaries

2.1. Storage codes for finite graphs.

Definition 2.1. [21, 28] Let $x \in \mathcal{Q}^n$ be a word over a finite alphabet $\mathcal{Q}$ and let $G = (V, E)$ be a finite graph with $|V| = \{1, \dots, n\}$ and a fixed ordering of the vertices. We say that x is assigned to G if there is a bijection $\{1, \dots, n\} \to V$ which places entries of x on the vertices. A *storage code* $\mathscr{C}$ on G is a collection of assignments of words such that for each $v \in V$ and every $x \in \mathscr{C}$, the value x_v is a function of $\{x_u, u \in \mathcal{N}(v)\}$, where $\mathcal{N}(v) := \{u : (v, u) \in E(G)\}$ is the vertex neighborhood of v in G.

We briefly mention the known results for storage codes on finite graphs as defined in the opening paragraph of the paper. Let G be a graph with n vertices and let $\mathcal{R}_q(\mathscr{C}) := \frac{1}{n} \log_q |\mathscr{C}|$ be the rate of a q-ary code $\mathscr{C}$ on G. In this case, the recovery set of a vertex v is $\mathcal{N}(v)$, and it depends on v. We denote the largest attainable rate of a storage code on G by $\mathsf{cap}(G) := \sup_q \mathcal{R}_q(\mathscr{C})$.

There are several ways of constructing storage codes with large rate. The most well-known one is the edge-to-vertex construction: given a (d-regular) graph, place a q-ary symbol on every edge and assign each vertex a d-vector of symbols written on the edges incident to it (again we assume an ordering of the edges). The size of the code is $(q^d)^{n/2}$, resulting in the rate value $1/2$ irrespective of the value of q. Here is an example with $q = d = 2$:



This method extends to the case when every vertex is incident to the same number of cliques. For instance, if this number is one, then the graph can be partitioned into k-cliques. To construct a code, we put a single parity symbol on every clique and distribute the symbols of the parity to the vertices that form it, resulting in rate $\mathcal{R} = (k-1)/k$. A further extension, known as *clique covering* [4], states that

$$\mathsf{cap}(G) \geq 1 - \alpha(G)/n, \tag{1}$$

where $\alpha(G)$ is the smallest size of a clique covering in G. Among other general constructions, we mention the *matching construction*, which yields

$$\mathsf{cap}(G) \geq M(G)/n,$$

where $M(G)$ is the size of the largest matching in G.

Turning to upper bounds, we note a result in [4], Theorem 3 (also [21], Lemma 9), which states that

$$\mathsf{cap}(G) \leq 1 - \gamma(G)/n, \tag{2}$$

where $\gamma(G)$ is the independence number of G, i.e., the size of the largest independent set of vertices (in other words, the size of the smallest vertex cover of G).

Given a graph G, we may sometimes want to look at neighbors at distance more than one from the failed vertex to recover its value, such as the set R discussed in Def. 4.1. Effectively, this changes the connectivity of the graph, so while we keep the same set of vertices V, the edges are now drawn according to where the failed vertex collects the data for its recovery. Assuming that the recovery region can be defined consistently for all the vertices, we will use the notation G_R in our discussion of storage codes for this case. This agreement will be used in particular when we address the two-dimensional grid $\mathbb{Z} \times \mathbb{Z}$ below.

Since R does not have to be symmetric, the graph G_R generally is directed (as is often the case in index coding [4, 3]). In this case, bound (2) affords a generalization, which we proceed to describe. For a subset of vertices $U \subseteq V$ in a graph $G = (V, E)$, denote by $G(U)$ the induced subgraph. A directed graph with no directed cycles is called a *directed acyclic graph* (DAG). It is known that a graph is a DAG if and only if it can be topologically ordered [19, p. 258], i.e., there is a numbering of the vertices such that the tail of every arc is smaller than the head. A set of vertices $S \subseteq V$ in a graph G is called a *DAG set* if the subgraph induced by S is a DAG. With this preparation, the following *maximum acyclic induced subgraph (MAIS) bound* is true; see [4], Theorem 3, or [3], Sec. 5.1.

Theorem 2.1. *Let $G = G(V, E)$ be a graph and let $\delta(G)$ be the size of the largest DAG set in it. Then,*

$$\mathsf{cap}(G) \leq 1 - \delta(G)/n. \tag{3}$$

Proof. Let $S \subset V$ be a DAG set. We argue that the values stored on S are uniquely determined by the values stored on $V \setminus S$. First, consider the topological ordering of S, all the outgoing edges of the last ordered $s \in S$ end in $V \setminus S$. Therefore, s can be recovered by $V \setminus S$. Then we can recover the second last vertex of S by s and $V \setminus S$. Proceeding like this, we are able to recover all values stored on S. Taking S to be the largest DAG set in G, we have $\mathsf{cap}_q(G) \leq 1 - \delta(G)/n$ for all $q \geq 2$. □

The bounds (2) and (3) are tight and can be achieved by clique partition codes. A *clique partition* of a graph G is a partition of the graph into subsets such that the induced graph by every subset is a clique.

Other upper bounds on the rate of storage codes are found in [22] for the symmetric case and in [3] for the general case. In this paper we will need a *linear programming bound* for the capacity of storage codes proved in [22] (its statement is somewhat technical and is given in Appendix A). We note that the authors of [22] also found some families of codes that achieve it.

2.2. **Capacity for finite graphs and the code-anticode bound.** Given a finite graph $G(V, E)$, we define the graphical distance $\rho(u, v)$ as the length of the shortest path in G between u and v. A code in G is a subset $\mathscr{C} \subset V$, and it is said to have minimum distance d if $\rho(u, v) \geq d$ for all pairs of distinct vertices $u, v \in \mathscr{C}$, or, in other words, if for any two distinct code vertices $u, v \in \mathscr{C}$ the balls $\mathcal{B}_r(u)$ and $\mathcal{B}_r(v)$ of radius $r = \lfloor (d-1)/2 \rfloor$ are disjoint. A code $\mathscr{C} \subset V$ is called an r-covering code if $\bigcup_{v \in \mathscr{C}} \mathcal{B}_r(v) = V$. We denote by $C(G, r)$ the smallest size of an r-covering code in the graph G.

Suppose that we are given a finite set $V, |V| = n$ and a subset $R(v) \subset V \setminus \{v\}$ that serves the recovery region for the vertex v. In this section the recovery regions will be given by balls of a given radius r in the metric ρ, the same for every vertex $v \in V$. We construct a graph $G_r = G(V, E)$ by connecting each vertex v with all the vertices u such that $1 \leq \rho(v, u) \leq r$. For instance, in Sec. 5 below, V will be an $n \times n$ region of $\mathbb{Z}^2$ and $R(v) = \mathcal{B}_r(v) \setminus \{v\}$, where $\mathcal{B}_r(\cdot)$ is a ball of radius r in some metric on $\mathbb{Z}^2$ (we focus on the l_1 and l_∞ distances).

Denote by $A(G; r+1)$ the size of the largest code in G with minimum distance at least $r+1$. Any such a code is an independent set in G_r, and thus by (2)

$$\mathsf{cap}(G_r) \leq 1 - \frac{1}{n}\gamma(G_r) = 1 - \frac{1}{n}A(G; r+1). \tag{4}$$

This gives an interpretation of the bound (2) in coding-theoretic terms. Furthermore, every ball of radius $\lfloor r/2 \rfloor$ in G is a clique in the graph G_r, and thus, any $\lfloor r/2 \rfloor$-covering code in G can form a clique covering of the graph G_r. As above, let $\alpha(G_r)$ be the number of cliques in the smallest covering. From (1) we obtain that

$$\mathsf{cap}(G_r) \geq 1 - \frac{1}{n}\alpha(G_r) \geq 1 - \frac{1}{n}C\left(G; \left\lfloor \frac{r}{2} \right\rfloor\right).$$

Assume the number of vertices in the ball of radius $\lfloor (r-1)/2 \rfloor$ in G does not depend on the center, and denote this number by $B_G(r)$. There are general conditions for this to hold, for instance if the graph G is arc-transitive. Then the *sphere packing bound* implies that $A(G; r+1) \leq \frac{n}{B_G(\lfloor (r-1)/2 \rfloor)}$. If r is even and there exists a perfect $r/2$-error-correcting code then $A(G; r+1) = C(G; r/2) = \frac{n}{B_G(\lfloor (r-1)/2 \rfloor)}$ and so

$$\mathsf{cap}(G_r) = 1 - \frac{1}{n}A(G; r+1) = 1 - \frac{1}{B_G(\lfloor (r-1)/2 \rfloor)}.$$

These relations can be used to derive capacity bounds for graphs. Below we use an extension of the sphere-packing bound, known as the *code-anticode bound*, to derive exact values of capacity for some recoverable systems in $\mathbb{Z}^2$. A subset of vertices in G with diameter D is called an *anticode*. For instance, a ball of radius τ is an anticode with diameter 2τ. It is known that in many cases the largest size of an anticode of even diameter D is achieved by a ball of radius $D/2$. For odd values of D the largest anticode is usually constructed by taking a union of two balls of radius $(D-1)/2$ whose centers are adjacent in G.

Delsarte [11, Thm.3.9] proved that if G contains a code $\mathscr{C}$ with minimum distance $r+1$ and an anticode $\mathscr{D}$ of diameter r, *and G is distance-regular*, then

$$|\mathscr{C}||\mathscr{D}| \leq n. \tag{5}$$

Taking $\mathscr{D}$ a ball of radius $\lfloor (r-1)/2 \rfloor$ recovers the sphere-packing bound. The condition of distance regularity was relaxed in [1, 13]. In particular, [1, Thm. 1′] implies that (5) holds true as long as G admits a transitive automorphism group. A code that satisfies the code-anticode bound with equality is called *diameter perfect*. The existence of diameter perfect codes is in general a difficult question; see e.g., [29, 14] for recent references. In our examples, diameter perfect codes will exist, and they will also generate a tiling of the graph with congruent copies of the corresponding anticode.

We continue with the following general claim.

Theorem 2.2. *Suppose that G has a transitive automorphism group and contains a diameter perfect code. Then,*

$$\mathsf{cap}(G_r) = 1 - \frac{1}{D_G(r)}, \tag{6}$$

where $D_G(r)$ is the largest possible size of an anticode in G of diameter r.

Proof. An anticode $\mathscr{D}$ of diameter r forms a clique in the graph G_r because the recovery region of every vertex in $\mathscr{D}$ includes all the other vertices in $\mathscr{D}$. Thus, by assumption, there is a clique covering of G_r, and if the anticodes are of the largest possible size, this clique covering is a smallest one. Then from (1) we conclude that $\mathsf{cap}(G_r) \geq 1 - \frac{1}{D_G(r)}$.

Further, from (5) we have $A(G; r+1) \leq n/D_G(r)$, and since G contains a diameter perfect code, this is in fact an equality. Then (4) implies that $\mathsf{cap}(G_r) \leq 1 - \frac{1}{D_G(r)}$, completing the proof. □

As an example, consider a *discrete torus*, i.e., a graph $\mathcal{T}_n$ on the vertex set $V = [n] \times [n]$ with an edge between (i_1, j_1) and (i_2, j_2) whenever $(i_1 - i_2, j_1 - j_2)$ equals one of $(1,0), (0,1), (-1,0), (0,-1)$ modulo n. Consider storage codes on $\mathcal{T}_n$ based on the recovery sets formed by the entire rows and columns (circles) on the torus. In other words, we take the recovery set in the form $R := (0,*) \cup (*,0) \backslash (0,0)$ where the unspecified coordinates are allowed to vary over the entire set $\mathbb{Z}_n$. The graph $(\mathcal{T}_n)_R$ representing the system is obtained from $\mathcal{T}_n$ by connecting all pairs of vertices whose coordinates are identical in either the first or the second position.

Theorem 2.3. *For $n \geq 3$ the storage capacity of the discrete torus $G := (\mathcal{T}_n)_R$ is*

$$\mathsf{cap}(G) = 1 - \frac{1}{n}. \tag{7}$$

Proof. Notice first that for any two vertices (i_1, j_1), (i_2, j_2), there is an edge between (i_1, j_1) and (i_2, j_2) if and only if $d_H((i_1, j_1), (i_2, j_2)) = 1$, where d_H denotes the Hamming distance. Placing a parity constraint on every row of $\mathcal{T}_n$ yields a code of rate $1 - 1/n$, proving a lower bound in (7). In order to show that it is also an upper bound, first observe that the automorphism

group of G acts transitively on it. On account of Theorem 2.2, to complete the proof it suffices to show that $D_G(1) = n$. In words, we want to show that the largest anticode of diameter 1 in the graph G under the Hamming metric is of size n, which is immediate. □

Remark 1. Using the results of Theorem 5.2, we can also find capacity of the torus with recovery region defined by the l_1 or l_∞ distance.

3. Interleaving Structures

In this section, we introduce a way to construct new storage codes for finite graphs by interleaving existing ones. Let $\mathscr{C}$ be a storage code on a finite graph G over the alphabet Q. Let $x^1, \ldots, x^{ks}$ be arbitrary ks codewords from $\mathscr{C}$, where $k, s \in \mathbb{N}$. Roughly speaking, the interleaving operation maps these ks codewords to $(Q^k)^{ns}$, and the set of all interleaved codewords forms a storage code on a new graph, defined below as a part of the code construction. Denote by $\bar{\mathscr{C}}$ the interleaved code and $\bar{G}$ the corresponding graph. We will define the interleaving operation such that $\mathcal{R}_{q^k}(\bar{\mathscr{C}}) = \mathcal{R}_q(\mathscr{C})$, thus $\mathsf{cap}_{q^k}(\bar{G}) \geq \mathsf{cap}_q(G)$ holds. Moreover, if $\mathcal{R}_q(\mathscr{C})$ meets the MAIS bounds (3) or the linear programming bound of Mazumadar et al. [22], then we have $\mathcal{R}_{q^k}(\bar{\mathscr{C}}) = \mathsf{cap}(\bar{G})$, enabling one to construct optimal storage codes for a wide range of parameters.

3.1. **Interleaving construction.**

Definition 3.1. Let $\mathcal{M}$ be a set of $k \times s$ matrices ($s \geq k$) over $\mathbb{N}$. We say that $\mathcal{M}$ forms a family of orthogonal partitions of the set of integers $\{1, \ldots, ks\}$ if

(1) Every $A \in \mathcal{M}$ contains every element of $\{1, \ldots, ks\}$ exactly once, thus, the columns of A form a partition of $\{1, \ldots, ks\}$.
(2) For every pair of distinct matrices $A, B \in \mathcal{M}$, every column of A has common entries with k columns of B, and every such pair of intersecting columns has exactly one common element.

We call (k, s) the shape and $|\mathcal{M}|$ the size of the family of orthogonal partitions, respectively.

Example 1. Let $k = 3, s = 5$. The following set of matrices forms a family of orthogonal partitions of the set $\{1, 2, \ldots, 15\}$:

$$\begin{bmatrix} 1 & 4 & 5 & 6 & 7 \\ 2 & 10 & 8 & 9 & 11 \\ 3 & 14 & 13 & 15 & 12 \end{bmatrix}, \begin{bmatrix} 1 & 2 & 3 & 4 & 6 \\ 8 & 5 & 13 & 11 & 10 \\ 9 & 7 & 14 & 15 & 12 \end{bmatrix}, \begin{bmatrix} 1 & 2 & 3 & 4 & 7 \\ 10 & 13 & 5 & 8 & 9 \\ 11 & 15 & 6 & 12 & 14 \end{bmatrix}, \begin{bmatrix} 1 & 2 & 3 & 6 & 7 \\ 4 & 12 & 9 & 11 & 8 \\ 5 & 14 & 10 & 13 & 15 \end{bmatrix},$$
$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 8 & 12 & 9 & 11 \\ 7 & 10 & 15 & 13 & 14 \end{bmatrix}, \begin{bmatrix} 1 & 2 & 3 & 5 & 6 \\ 12 & 9 & 4 & 10 & 8 \\ 13 & 11 & 7 & 15 & 14 \end{bmatrix}, \begin{bmatrix} 1 & 2 & 3 & 5 & 7 \\ 14 & 4 & 8 & 9 & 10 \\ 15 & 6 & 11 & 12 & 13 \end{bmatrix}.$$

For instance, the second column of the first matrix interests with columns 3,4, and 5 of the second matrix, etc.

Families of orthogonal partitions can be constructed relying on resolvable designs [10], which we define next. Let $\mathcal{B}$ be a collection of k-subsets of a finite set V. We call the elements of V and $\mathcal{B}$ points and blocks, respectively. The pair $(V, \mathcal{B})$ is called an r-(v, k, λ) block design if every r-subset of V is contained in exactly λ blocks. A set of blocks is called a parallel class if they form a partition of V. Finally, a block design is called resolvable if its set of blocks $\mathcal{B}$ can be partitioned into parallel classes.

Proposition 3.1. *A* 2-$(v, k, 1)$ *resolvable design defines a family of orthogonal partitions of shape* (k, s) *and size* $(v-1)/(k-1)$*, where* $s = v/k$.

Proof. Given a 2-$(v,k,1)$ resolvable design, each parallel class yields a matrix in the family of orthogonal partitions, whose columns are the blocks in the class[1]. Therefore, every matrix contains all the points exactly once, and two columns from different matrices are either disjoint or intersect on one point. Indeed, if two columns intersect on two points, this would imply that a 2-subset is contained in two blocks, which is a contradiction. □

By Proposition 3.1, the existence of 2-$(v,k,1)$ resolvable design implies the existence of orthogonal partition families. The necessary conditions for the existence of 2-$(v,k,1)$ resolvable designs are $k|v$ and $(k-1)|(v-1)$. If v and k are both powers of the same prime, then the necessary conditions are also sufficient [10, Theorem 7.10]. In particular, for every $v \equiv 3 \mod 6$ there exists a family of 2-$(v,3,1)$ resolvable designs called Kirkman triple systems (Example 1 is obtained from a Kirkman triple system with $v = 15$). Therefore, resolvable designs give us a rich collection of orthogonal families[2].

Now we will present a way of obtaining new storage codes from known codes. Given a code $\mathscr{C}$ on a graph $G = (V,E)$ and a family $\mathcal{M}$ of orthogonal partitions of shape (k,s), we will first construct a new graph $\bar{G} = (\bar{V},\bar{E})$. Fix a coloring of G in c colors such that adjacent vertices are assigned different colors. Pick c matrices from $\mathcal{M}$ and label them with colors $M_1,\dots,M_c$. Let $\bar{V} = \{1,\dots,n\} \times \{1,\dots,s\}$, and $((t,\mu),(t',\mu')) \in \bar{E}$ if and only if $(t,t') \in E$, and column μ of $M_{c(t)}$ intersects column μ' of $M_{c(t')}$, where $c(\cdot)$ denotes the color of the vertex. Note that each vertex $t \in V$ corresponds to an independent set $\bar{t} = \{(t,1),\dots,(t,s)\} \in \bar{V}$, and there are edges between vertices in $\bar{t}$ and $\bar{t}'$ only if $(t,t') \in E$. This construction works for both undirected and directed graphs G.

Next, given a storage code $\mathscr{C}$ on a finite graph $G = (V,E)$ over the alphabet $\mathcal{Q}$, $|\mathcal{Q}| = q$, and $V = \{1,\dots,n\}$, we define an interleaving procedure that produces a storage code $\bar{\mathscr{C}}$ on $\bar{G}$ over $(\mathcal{Q}^k)^{ns}$.

(1) Choose codewords $x^1, x^2, \dots, x^{ks}$ from $\mathscr{C}$ (not necessarily distinct). For each $x^\lambda \in \mathscr{C}$, $\lambda = 1,\dots,ks$, denote by x_v^λ the symbol stored on the vertex $v \in V$.
(2) For each vertex $t \in V$, denote by $r = c(t)$ its color, and form a $k \times s$ matrix such that entry (i,j) is $x_t^{M_r(i,j)}$. In other words, we arrange $x_t^1,\dots,x_t^{ks}$ in a $k \times s$ matrix X_t according to $M_r \in \mathcal{M}$, namely, the i,j-th entry is $x_t^{M_r(i,j)}$.
(3) A codeword $\bar{x} \in (\mathcal{Q}^k)^{ns}$ is obtained by concatenating the columns of the matrices $X_1, X_2, \dots, X_n$ in some fixed order. Specifically, for $\mu \in \{1,\dots,s\}$ and $t \in \{1,\dots,n\}$ we assign column μ of X_t to the vertex (t,μ).
(4) The collection of codewords $\bar{x}$ constructed in the previous steps forms the code $\bar{\mathscr{C}}$.

Proposition 3.2. *The interleaved code $\bar{\mathscr{C}}$ is a storage code on $\bar{G}$.*

Proof. Suppose that node (t,μ) is erased, and denote by $(x_t^{\lambda_1},\dots,x_t^{\lambda_k})^T$ the column stored on it, where $x^{\lambda_1},\dots,x^{\lambda_k} \in C$ are the codewords that were interleaved to define the codeword $\bar{x}$. Note that $x_t^{\lambda_j}$ is a function of $\{x_{t'}^{\lambda_j}, t' \in \mathcal{N}(t)\}$, and $x_{t'}^{\lambda_j}$ is stored on exactly one of the nodes $\bar{t}'$, and by definition of orthogonal partitions, $x_{t'}^{\lambda_j}, x_{t'}^{\lambda_{j'}}$ are stored on different vertices if $j \neq j'$. In a nutshell, (t,μ) can be recovered from the values $\bigcup_{j=1}^k \{x_{t'}^{\lambda_j}, t' \in \mathcal{N}(t)\}$, which are stored on exactly $k|\mathcal{N}(t)|$ vertices of $\bar{G}$. □

[1] We only require each column to contain all the elements from the block and do not impose any ordering.

[2] Every 2-$(v,k,1)$ resolvable design yields $(k!)^{\frac{v(v-1)}{k(k-1)}}(s!)^{\frac{v-1}{k-1}}$ orthogonal families. To see this, note that there are $v(v-1)/k(k-1)$ blocks and $(v-1)/(k-1)$ parallel classes, and we can rearrange elements in each column and permute the columns of each matrix.

Remark 2. It is straightforward to check that the described method enables us to interleave codewords from different storage codes on G by the same method. Namely, let $\mathscr{C}_1, \dots, \mathscr{C}_{ks}$ be ks storage codes on G, and choose codewords $x^i \in \mathscr{C}_i$ for each $i = 1, 2, \dots, ks$. They can be used in the above procedure to define a new code. At the same time, if our goal is to construct large-size codes, then the construction should rely on codes of the largest known size, for instance, copies of the same code.

Example 2 (Example of interleaving). Let G be a triangle and $\mathscr{C}$ be a linear parity-check storage code over $\mathbb{F}_3$.

(1) Color G by 3 colors and let

$$\mathcal{M} = \left\{ M_1 = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}, M_2 = \begin{bmatrix} 1 & 2 & 6 \\ 5 & 3 & 4 \end{bmatrix}, M_3 = \begin{bmatrix} 1 & 5 & 6 \\ 4 & 2 & 3 \end{bmatrix} \right\}.$$

(2) Pick 6 codewords from $\mathscr{C}$, denote them by $x^1, \dots, x^6$, and arrange them into matrices X_1, X_2, X_3 relying on the orthogonal partitions given by M_1, M_2, M_3. We obtain

$$X_1 = \begin{bmatrix} x_1^1 & x_1^3 & x_1^5 \\ x_1^2 & x_1^4 & x_1^6 \end{bmatrix}, X_2 = \begin{bmatrix} x_2^1 & x_2^2 & x_2^6 \\ x_2^5 & x_2^3 & x_2^4 \end{bmatrix} X_3 = \begin{bmatrix} x_3^1 & x_3^5 & x_3^6 \\ x_3^4 & x_3^2 & x_3^3 \end{bmatrix},$$

and

$$\bar{x} = \begin{pmatrix} x_1^1 & x_1^3 & x_1^5 & x_2^1 & x_2^2 & x_2^6 & x_3^1 & x_3^5 & x_3^6 \\ x_1^2 & x_1^4 & x_1^6 & x_2^5 & x_2^3 & x_2^4 & x_3^4 & x_3^2 & x_3^3 \end{pmatrix}.$$

For instance, choosing $\{x^1 = 111, x^2 = 222, x^3 = 000, x^4 = 120, x^5 = 012, x^6 = 102\} \subset \mathscr{C}$, we obtain

$$\bar{x} = \begin{pmatrix} 1 & 0 & 0 & 1 & 2 & 0 & 1 & 2 & 2 \\ 2 & 1 & 1 & 1 & 2 & 2 & 0 & 0 & 0 \end{pmatrix}.$$

The corresponding graph $\bar{G}$ defined on $[3] \times [3]$ is shown in the figure.

3.2. **The rate of interleaved codes.** An immediate observation is as follows.

Proposition 3.3. *We have* $\mathcal{R}_{q^k}(\bar{\mathscr{C}}) = \mathcal{R}_q(\mathscr{C})$*, and thus* $\mathsf{cap}_{q^k}(\bar{G}) \geq \mathsf{cap}_q(G)$.

Proof. The rate of $\bar{\mathscr{C}}$ satisfies

$$\mathcal{R}_{q^k}(\bar{\mathscr{C}}) = \frac{1}{ns} \log_{q^k} |\mathscr{C}|^{ks} = \frac{1}{n} \log_q |\mathscr{C}| = \mathcal{R}_q(\mathscr{C}).$$

The second claim is obvious. □

Remark 3. Let $\tilde{G} = (\tilde{V}, \tilde{E})$ be a graph such that $\tilde{V} = \{1, \dots, n\} \times \{1, \dots, s\}$ and $((t, \mu), (t', \mu')) \in \bar{E}$ *if and only if* $(t, t') \in E$. Then storage codes on $\tilde{G}$ over $\mathcal{Q}$ are essentially the same as storage codes on G over Q^s, and thus, $\mathsf{cap}_q(\tilde{G}) = \mathsf{cap}_{q^s}(G)$. Since $\bar{G} = \tilde{G}$ if $k = s$, we focus on the case $k < s$.

Recall that $\delta(G)$ denotes the size of the largest DAG in G.

Proposition 3.4. *Let* $\mathscr{C}$ *be a storage code on* G *with* $\mathcal{R}_q(\mathscr{C}) = \mathsf{cap}_q(G)$*. If* $\mathsf{cap}_q(G) = 1 - \delta(G)/n$*, then the interleaved code* $\bar{\mathscr{C}}$ *achieves the maximum rate on the corresponding graph* $\bar{G}$*, and* $\mathsf{cap}_{q^k}(\bar{G}) = \mathsf{cap}(G)$.

Proof. We first show that $\delta(\bar{G}) \geq s\delta(G)$. Let $S \subset V$ be the largest DAG set of G, and $\bar{S} := \bigcup_{v \in S} \bar{v}$ (recall that $\bar{v} = (v, \sigma)$, where σ runs over $\{1, \dots, s\}$). It is straightforward to check that $\bar{S}$ is also a DAG in $\bar{V}$ of size $s\delta(G)$. Indeed, if $(t_1, \mu_1), \dots, (t_m, \mu_m), (t_1, \mu_1)$ is a directed

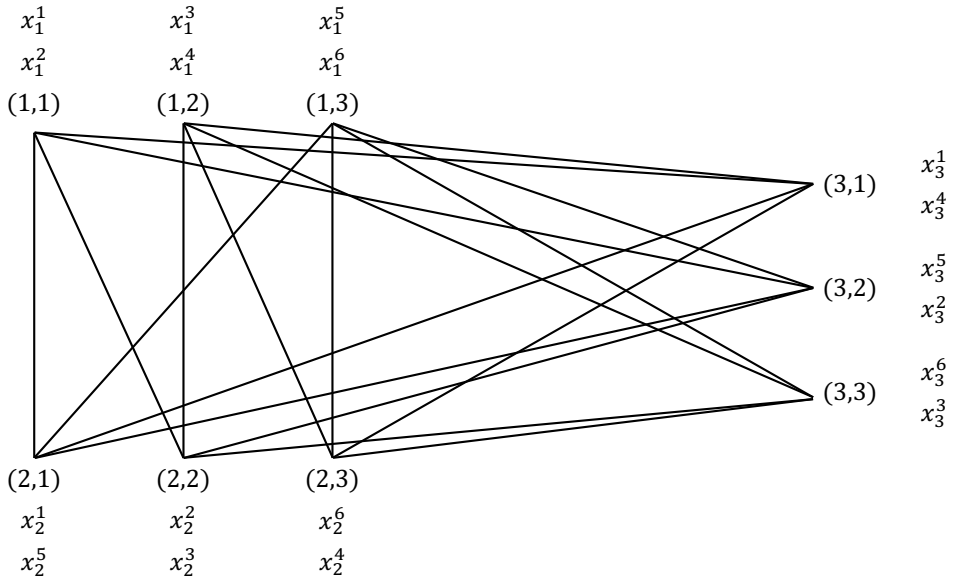


FIGURE 1. The figure shows the graph $\bar{G}$ in the example. We have $n = 3, k = 2, s = 3$, and $c = 3$. The graph has 9 vertices connected as shown. Next to each vertex we show the pair of letters stored in it.

cycle in $\bar{S}$, then $t_1, \dots, t_m, t_1$ is a directed cycle in S. Therefore, $\delta(\bar{G}) \geq |\bar{S}| = s|S| = s\delta(G)$. Now by the MAIS bound (3),

$$\mathsf{cap}_{q^k}(\bar{G}) \leq 1 - \frac{\delta(\bar{G})}{ns} \leq 1 - \frac{|\bar{S}|}{ns} = 1 - \frac{\delta(G)}{n} = \mathsf{cap}_q(G) = \mathsf{cap}(G).$$

On the other hand, by Proposition 3.3, we have $\mathsf{cap}_{q^k}(\bar{G}) \geq \mathsf{cap}_q(G) = \mathsf{cap}(G)$. □

We remark that given a largest DAG set in G, we can easily identify a largest DAG in $\bar{G}$.

Corollary 3.5. *Let $G = (V, E)$ be a graph with a storage code $\mathscr{C}$ that satisfies $\mathcal{R}_q(\mathscr{C}) = 1-\delta(G)/n$. We have $\delta(\bar{G}) = s\delta(\bar{G})$, where $\bar{G}$ is the graph defined by the interleaving procedure. In addition, if $S \subset V$ is a largest DAG of G, then the set $\bar{S} = \bigcup_{v\in S} \bar{v}$ is a largest DAG in $\bar{G}$.*

Proof. From the proof of Proposition 3.4, we know that $\bar{S}$ is a DAG in $\bar{G}$ and $\delta(\bar{G}) \geq s\delta(G)$. On the other hand, by Propositions 3.3 and 3.4, we have

$$1 - \frac{\delta(G)}{n} = \mathsf{cap}_q(G) = \mathcal{R}_q(C) = \mathcal{R}_{q^k}(\bar{\mathscr{C}}) = \mathsf{cap}_{q^k}(\bar{G}) \leq 1 - \frac{\delta(\bar{G})}{ns},$$

which implies that $\delta(\bar{G}) \leq s\delta(G)$. Hence, $\delta(\bar{G}) = s\delta(G)$, and $\bar{S}$ is a largest DAG set. □

In particular, this claim is true if S is an independent set in G.

Corollary 3.6. *Let $\mathscr{C}$ be a storage code on G with $\mathcal{R}_q(\mathscr{C}) = \mathsf{cap}_q(G)$. If $\mathsf{cap}_q(G) = 1 - \gamma(G)/n$, then the interleaved code $\bar{\mathscr{C}}$ achieves the maximum rate on the corresponding graph $\bar{G}$, and $\mathsf{cap}_{q^k}(\bar{G}) = \mathsf{cap}(G)$. Further, if $S \subset G$ is the largest independent set of G, then $\bar{S} = \bigcup_{v\in S} \bar{v}$ is the largest independent set of $\bar{G}$.*

A similar claim holds for the codes that achieve the linear programming bound of [22].

Proposition 3.7. *Let $\mathscr{C}$ be a storage code on $G = (V, E)$ with $\mathcal{R}_q(\mathscr{C}) = \mathsf{cap}_q(G)$. If $\mathsf{cap}_q(G)$ achieves the LP bound of Theorem A.1, then the interleaving code $\bar{\mathscr{C}}$ has the maximum possible rate on the corresponding graph $\bar{G} = (\bar{V}, \bar{E})$, and $\mathsf{cap}_{q^k}(\bar{G}) = \mathsf{cap}(G)$, where $\bar{G}$ is defined by the interleaving structure.*

The proof is given in Appendix A.

Note that generally, for any subgraph $G' \subset \tilde{G}$ defined on the vertex set $\tilde{V}$, there is an obvious relation between the capacities, $\mathsf{cap}(G') \leq \mathsf{cap}(G)$. Of course, for most subgraphs we do not have a means of constructing good codes; however, for subgraphs of the form $\bar{G}$ derived from the interleaving construction, we have a way of obtaining optimally sized codes as described above.

3.3. **Some optimal storage codes.** The interleaving construction relies on a good seed code. In this section we list some known code families that can be used to obtain new optimal storage codes by interleaving, and working out the details of the construction for one of them. Below we use the notation $[n] = \{0, 1, \ldots, n-1\}$.

Example 3. Let l, r be positive integers, and $R = \{1 + [r]\} \cup \{-1 - [l]\}$, and let $m = \min\{l, r\}$. In Proposition 4.2 below we find the capacity of a recoverable system X_R defined on the (infinite) graph $\mathbb{Z}_R$. Taking a finite subgraph $G = \mathbb{Z}_R \cap [n]$, where $(m+1)|n$, we obtain an optimal storage code with $\mathcal{R}(\mathscr{C}) = m/(m+1)$. The code can be obtained from the clique partition construction described above (1), and since $\{i(m+1) : 0 \leq i < n/(m+1)\}$ is the largest DAG set in G, we obtain that $\mathsf{cap}(G) = 1 - \frac{n/(m+1)}{n} = m/(m+1)$.

Let us apply the interleaving construction for this code. First, we color G with colors $\{0, \ldots, l + r\}$ such that vertex t is colored with color $\tau := t \bmod (l + r + 1)$. Next, we choose a family of orthogonal partitions $\mathcal{M} = \{M_0, \ldots, M_{r+l}\}$ of shape (k, s) and define $\bar{G} = (\bar{V}, \bar{E})$ such that $\bar{V} = \{1, \ldots, n\} \times \{1, \ldots, s\}$ and $((t, \mu), (t', \mu')) \in \bar{E}$ if $(t, t') \in E$, and column μ of M_τ intersects column μ' of $M_{\tau'}$. Finally, the interleaved code is obtained by the procedure defined before Proposition 3.2, where $\mathscr{C}$ is the optimal clique partition code described in the previous paragraph.

Example 4 (Cycles). Let C_n be an undirected cycle of length n. If n is even, then the largest independent set of C_n has size $n/2$, so $\mathsf{cap}(C_n) \leq 1/2$ by (2). At the same time, the edge-to-vertex construction yields a storage code $\mathscr{C}$ of rate $1/2$ over an alphabet Q. If n is odd, then by the LP bound it can be shown that $\mathsf{cap}(C_n) \leq 1/2$. A code of rate $1/2$ can be obtained from the fractional matching construction [22].

Example 5. ([22, Theorem 10]) Let C_k be a cycle of length $k > 3$ and let B be a bipartite graph. Then $\mathsf{cap}(C_k \boxtimes B) = \frac{1}{2}$, where $\boxtimes$ is the Cartesian product of graphs[3]. The fractional matching construction gives optimal codes.

Example 6. ([22, Theorem 11]) Let G be a cycle with chords whose endpoints are at least distance 4 apart on the cycle. We have $\mathsf{cap}(G) = \frac{1}{2}$, and again, the fractional matching construction gives optimal codes.

In the last two examples, optimality is proved by relying on the linear programming bound.

[3]The Cartesian product of two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is defined by $G = (V, E)$, where $V = V_1 \times V_2$; $((u, u'), (v, v')) \in E$ if and only if $u = v$ and $(u', v') \in E_2$ or $u' = v'$ and $(u, v) \in E_1$.

3.4. **Codes with partial recovery.** We say that a storage code $\mathscr{C}$ over alphabet Q^k has m levels of recovery if for every vertex v, by visiting a $1/m$ proportion vertices in $\mathcal{N}(v)$, we can recover a $1/m$ fraction of the symbols in v. Sometimes it may be desirable to have codes with multiple levels of recovery, since we do not need to visit all neighbors to recover only a part of the node. Consider the interleaved code, for any vertex $(t,\mu) \in \bar{V}$, let $(x_t^{\lambda_1},\dots,x_t^{\lambda_k})^T$ be the vector stored on it. Suppose that, to recover a subset of the coordinates of this vector, it suffices to download information from a part of the neighbors of (t,μ), and denote by $\mathcal{N}(x_t^{\lambda_j})$ the neighbors that determine the values $x_t^{\lambda_j}, j=1,\dots,k$. The interleaved construction relies on families of orthogonal partitions, and this ensures that the sets $\mathcal{N}(x_t^{\lambda_j}), j=1,\dots,k$ are disjoint and are of the same size. Therefore, our interleaving structure yields a storage code with k levels of recovery, and the rate is optimal if the seed code is optimal with respect to the MAIS bound or the LP bound.

3.5. **High-rate storage codes on triangle-free graphs.** It is easy to construct high-rate storage codes on graphs with many cliques, but constructing codes of rate $> 1/2$ on triangle-free graphs is a more difficult problem. Recently, [7] constructed infinite families of binary linear storage codes of rate approaching $3/4$. Following up on this research, two concurrent papers, [6] and [18], constructed families of binary linear storage codes of rate asymptotically approaching one. These constructions are based on Cayley graphs obtained as coset graphs of binary linear codes, and they are restricted to the binary alphabet. Here we wish to remark that the code families can be extended to alphabets of size 2^k by interleaving.

Let $\mathscr{C}$ be a binary linear storage code on a graph G, and let $\bar{\mathscr{C}}$ be the interleaved code on the corresponding graph $\bar{G}$. It is easy to see that if G is triangle-free then so is $\bar{G}$, and by Proposition 3.3, we have $\mathcal{R}_{2^k}(\bar{\mathscr{C}}) = \mathcal{R}_2(\mathscr{C})$. Therefore, interleaving the codes from [6] or [18] we obtain new families of storage codes with asymptotically unit rate over the alphabet $\mathbb{Z}_2^k$.

4. Recoverable Systems

4.1. **Recoverable systems: Definitions.** The following definition formalizes the concept of storage codes discussed above for the case of $V=\mathbb{Z}^d, d\geq 1$.

Definition 4.1. (Recoverable systems) Let $R\subset\mathbb{Z}\backslash\{0\}$ be a finite subset of integers ordered in a natural way. An *R-recoverable system* $X=X_R$ on $\mathbb{Z}$ is a set of bi-infinite sequences $x\in\mathcal{Q}^{\mathbb{Z}}$ such that for any $i\in\mathbb{Z}$ there is a function $f_i:\mathcal{Q}^{|R|}\to\mathcal{Q}$ such that $x_i=f_i(x_{i+R})$ for any $x\in X$.

More generally, let $R\subset\mathbb{Z}^d\backslash\{0\}$ be a finite subset together with some ordering. An *R-recoverable system* $X=X_R$ on $\mathbb{Z}^d$ is a set of letter assignments (d-dimensional words) x such that for any vertex $v\in\mathbb{Z}^d$ there is a function $f_v:\mathcal{Q}^{|R|}\to\mathcal{Q}$ such that $x_v=f_v(x_{v+R})$ for any $x\in X$. Here $v+R=\{v+z : z\in R\}$ is a translation of v by the recovery neighborhood (set) R.

This definition is close to the definition of storage codes [21] which also allows the dependency of the recovery function on $v\in V$ and also (implicitly) assumes an ordering of the vertices in V. Here we wish to note an important point: by defining the recovery set R we effectively introduce an edge (possibly, a directed one) between v and every vertex in $v+R$. Thus, we speak of R-recoverable systems on $\mathbb{Z}^d$ with edges defined by R, and denote the graphs that emerge in this way by $\mathbb{Z}^d_R$. If $d=1$, we omit it from the notation.

Example 7. Take $V=\mathbb{Z}$ and $R=\{-1,1\}$. Assume that $x_{2i+1}=x_{2i}$ for all i. Then we have $f_i(x_{i-1},x_{i+1})=x_{i+(-1)^i}$, that is, $f_i(x_{i-1},x_{i+1})=x_{i-1}$ or x_{i+1} depending on whether i is odd or even.

Definition 4.2. (CAPACITY, ONE-DIMENSIONAL) Let $X = X_R$ be a one-dimensional R-recoverable system. For $n \geq 0$, denote by $B_n(X)$ the restriction of the words in X to the set $[n] := \{0, 1, \ldots, n-1\}$, i.e., $B_n(X) = \{x_{[n]} : x \in X\}$. The *rate* of X is defined as

$$\mathcal{R}(X) = \mathcal{R}_q(X) := \limsup_{n\to\infty} \frac{1}{n} \log_q |B_n(X)|. \tag{8}$$

The dependence on q will be omitted when it plays no role.

Given a recovery set $R \subset \mathbb{Z}\backslash\{0\}$, we are interested in its largest attainable rate

$$\mathsf{cap}(\mathbb{Z}_R) := \sup_{X_R \subseteq \mathcal{Q}^{\mathbb{Z}}} \mathcal{R}(X_R) \tag{9}$$

of R-recoverable systems on $\mathbb{Z}$, calling it the capacity of the graph $\mathbb{Z}_R$.

Example 8. Consider the R-recoverable system from Example 7. Since every odd symbol is uniquely determined by the two adjacent symbols in even positions, we have $B_n(X) \leq q^{\lceil n/2 \rceil}$, and since every symbol can occur in the even positions, we have $B_n(X) \geq q^{\lfloor n/2 \rfloor}$. Thus, $\mathcal{R}(X) = \frac{1}{2}$.

Similarly, we define the capacity of multidimensional systems.

Definition 4.3. (CAPACITY, d DIMENSIONS) Let $R \subset \mathbb{Z}^d \setminus \{0\}$ be a finite set (the recovery set) and let $\mathcal{B}_{[n]^d}(X) = [n]^d$ be the d-dimensional cube. Given a recoverable system $X = X_R$, define its rate as

$$\mathcal{R}_q(X) = \limsup_{n\to\infty} \frac{1}{n^d} \log_q |\mathcal{B}_{[n]^d}(X)|,$$

and let $\mathsf{cap}(\mathbb{Z}_R^d) = \sup_{X_R \subseteq Q^{\mathbb{Z}^d}} \mathcal{R}(X_R)$ be the largest rate associated with the set R.

At this point one may wonder if the capacity value depends on our choice of cubes in this definition. For the invariant case we prove that the answer is no, although a rigorous treatment is somewhat technical, and we postpone it till Sec.4.3.

4.2. **Recoverable systems on $\mathbb{Z}$.** Bounds and constructions for finite graphs can be extended to infinite graphs such as $\mathbb{Z}^d$. In this section, we state a few easy results that exemplify the above methods for the case of $d = 1$. Analogous claims in higher dimensions are discussed in the next subsection.

Let $R \subset \mathbb{Z} \setminus \{0\}$ be a finite set and consider R-recoverable systems on the infinite graph $\mathbb{Z}_R$. For all $n > 0$ let us consider the subgraph $\mathbb{Z}_{R,n} := \mathbb{Z}_R \cap [n]$. To bound above the capacity $\mathsf{cap}(\mathbb{Z}_R)$ we first find $\mathsf{cap}(\mathbb{Z}_{R,n})$ and argue that these quantities are related. This follows from the observation that we can disregard the boundary effects because the proportion of points whose recovery regions do not fit in $[n]$ vanishes as n increases. Thus, we can place constant values on these points with a negligible effect on the rate of the code.

Lemma 4.1. *Let $R \subset \mathbb{Z} \setminus \{0\}$ be a finite set and $(c_n)_{n=1}^{\infty}$ an infinite sequence of real numbers such that $\mathsf{cap}(\mathbb{Z}_{R,n}) \leq c_n$ for all $n \geq 1$. Then, $\mathsf{cap}(\mathbb{Z}_R) \leq \limsup_{n\to\infty} c_n$.*

Proof. For a given $n \geq 1$, let A_n be the set of vertices in $\mathbb{Z}_{R,n}$ in which their neighborhood in the graph $\mathbb{Z}_R$ does not fully belong to the graph $\mathbb{Z}_{R,n}$, i.e., $A_n = \{m \in [n] : m + R \nsubseteq [n]\}$. Since $|R|$ is a constant independent of n, starting from some value of n we have $|A_n| \leq a$ for some absolute constant a.

Let X be an R-recoverable system and recall that $B_n(X)$ is its restriction to $[n]$. For a given vector $z = (x_i)_{i\in A_n} \in \mathcal{Q}^{|A_n|}$, let $B_n^z(X)$ be the set of words in $B_n(X)$ which match the values of z over the positions in A_n. The code $B_n^z(X)$ is a storage code over the graph $\mathbb{Z}_{R,n}$ and hence

$$n^{-1} \cdot \log_q |B_n^z(X)| \leq \mathsf{cap}(\mathbb{Z}_{R,n}) \leq c_n.$$

Since the inequality holds for all $z \in \mathcal{Q}^{|A_n|}$ and $|A_n| \leq a$, we conclude that

$$n^{-1} \cdot \log_q |B_n(X)| \leq c_n + a/n,$$

which verifies the statement of the lemma. □

According to Lemma 4.1, it is possible to calculate the capacity of several recovery sets R.

Proposition 4.2. *Let l, r be positive integers, let $R = \{1 + [r]\} \cup \{-1 - [l]\}$, and let $m = \min\{l, r\}$. Then,*

$$\mathsf{cap}(\mathbb{Z}_R) = \frac{m}{m+1}.$$

Proof. Assume without loss of generality that $r \leq l$, so $m = r$, and let $\mathbb{Z}_R$ be the graph describing the R-recoverable system. The upper bound follows by noting that for all $n > 1$ the set of vertices $\{k(r+1) : k \in \mathbb{Z}\} \cap [n]$ is a DAG set in the graph $\mathbb{Z}_{R,n}$ and thus

$$\mathsf{cap}(\mathbb{Z}_{R,n}) \leq 1 - \frac{\delta(\mathbb{Z}_{R,n})}{n} \leq 1 - \frac{\lceil \frac{n}{r+1} \rceil}{n} \leq \frac{r}{r+1}.$$

Together with Lemma 4.1 this implies that $\mathsf{cap}(\mathbb{Z}_R) \leq r/(r+1)$.

To show the reverse inequality we use the clique partition construction (1). Start with partitioning $\mathbb{Z}$ into segments of length $r+1$, setting $\mathbb{Z} = \bigcup_{k \in \mathbb{Z}} k \cdot [r+1]$. We aim to construct a code in which every vertex in the segment can be recovered from the other vertices in it. In other words, $\mathbb{Z}_R$ is a disjoint union of $(r+1)$-cliques, and the construction described above before (1) yields a code of rate $r/(r+1)$. □

In the case of *shift invariant* recoverable systems the upper bound of this proposition was derived in [12]; however, the authors of [12] stopped short of finding a matching construction under this assumption. The challenge in the shift invariant case is to find a recoverable system with the *same* recovery function for all symbols.

The next claim concerns recoverable systems in which the recovery functions use only two symbols for the recovery process, but those symbols are not necessarily adjacent to the symbol to be recovered.

Proposition 4.3. *Let l, r be positive integers and let $R = \{-l, r\}$. Then for any $q \geq 2$*

$$\mathsf{cap}(\mathbb{Z}_R) = \frac{\gcd(l, r)}{l + r}.$$

Proof. We use a similar technique to the one used in the previous proof. First, we present an upper bound on the capacity. Denote by $d := \gcd(l, r)$, $m = l + r$, and assume without loss of generality that $l \leq r$. For a sequence $x \in \mathcal{B}_n(X)$ for n large enough, we show how we can leave only the symbols in positions $i \in A := \{km + j \ : \ k \in \mathbb{N},\ j \in [d]\}$ since all the other symbols can be uniquely determined by the symbols in the positions of A. Notice that $x_l, \ldots, x_{l+d-1}$ can be recovered at first since $x_{[d]}, x_{m+[d]} \in A$. As a matter of fact, this can be done for every m-block in x, i.e., $x_{km+l+[d]}$ can be recovered for every $k \in [n/m]$. At the next step, it is possible to recover all the symbols $x_{2l+[d]}$ since during the previous step we recovered the symbols in positions $x_{l+[d]}$ and in $x_{2l+r+[d]}$. Again, this can be done in every m-block so it is possible to recover the symbols x_i for $i \in \{km + 2l + [d] \ : \ k \in \mathbb{N}\}$. We continue in the same way such that at the t-th step, we recover the set $x_{km+tl+[d]}$ for $k \in \mathbb{N}$. We are only left to show that the symbols in $x_d, x_{d+1}, \ldots, x_{l-1}$ are recovered. Since $d = \gcd(l, r)$, there exists t_1 such that $t_1 l = k_1 m + d$ for some k_1 which implies that $x_{km+d+[l]}$ are recovered.

The lower bound is given by the following encoding process. The information symbols are stored in positions $i \in \{km + [d] : k \in \mathbb{N}\}$. Then the parity symbols are added in the same order as above. □

Example 9. We demonstrate the encoding process for a recoverable system X_R with $R = \{-l, r\}$ over a three-letter alphabet. Take $l = 6, r = 4$, then $\gcd(4, 6) = 2$. We show the encoding process for positions $10, 11, \ldots, 19$, and note that similar steps are made for every other $(l + r)$-block. First, we put data symbols in positions $10k, 11k$ for all $k \in \mathbb{Z}$. The other symbols are chosen so as to satisfy the parity $x_i + x_{i-l} + x_{i+r} = 0$. To illustrate this, find $x_{14} = -(x_{10} + x_{20})$ and $x_{15} = -(x_{11} + x_{21})$, then find (x_{18}, x_{19}) from $(x_{14}, x_{15}; x_{24}, x_{25})$, etc.

Example 10. For the same set $R = \{-6, 4\}$, put a codeword of the length-5 repetition code on each of the mutually disjoint sets of consecutive symbols in the even positions and in the odd positions. For instance, we can take those sets to be $\{10k + 2j\}$ and $\{10k + 2j + 1\}$ with $0 \le j \le 4, k \in \mathbb{Z}$. Then clearly for every $i \in \mathbb{Z}$, the symbol x_i is a part of the codeword of the repetition code that includes either position $i - 6$ or position $i + 4$, and we can recover x_i by accessing one of those positions as appropriate.

Let $B \subset \mathbb{Z}_+$ be a finite or infinite set. We call a set $A \in \mathbb{Z}$ *B-avoiding* if its difference set $|A - A|$ is disjoint from B, i.e., $|b_1 - b_2| \notin B$ for all $b_1, b_2 \in A$. Let a_n be the size of the largest set $A_n \subseteq [n]$ that is B-avoiding and define $\mathcal{R}(B) = \limsup_{n\to\infty} a_n/n$. This quantity has been extensively studied in the literature, initially with B being the set of all whole squares [27] and later values of other polynomials; see, e.g., [25].

Proposition 4.4. *Let $0 < r_1 < r_2 < \cdots < r_s$ and $0 < l_1 < l_2 < \cdots < l_t$ be positive integers and $R = \{-l_t, \ldots, -l_2, -l_1, r_1, r_2, \ldots, r_s\}$. Then,*

$$\mathsf{cap}(\mathbb{Z}_R) \le 1 - \max\{\mathcal{R}(\{l_1, \ldots, l_t\}), \mathcal{R}(\{r_1, \ldots, r_s\})\}.$$

Proof. Assume without loss of generality that $\mathcal{R}(\{l_1, \ldots, l_t\}) \le \mathcal{R}(\{r_1, \ldots, r_s\})$ and let A_n be the largest subset of $[n]$ that is $\{r_1, r_2, \ldots, r_s\}$-avoiding. Then, A_n is a DAG set in the graph $\mathbb{Z}_{R,n}$ and thus $\mathsf{cap}(\mathbb{Z}_{R,n}) \le 1 - |A_n|/n$. Indeed, all edges in A_n are going left. The statement follows from Lemma 4.1. □

Consider for example the set $B_m = \{1, 2, 4, \ldots, 2^m\}$ for $m \ge 1$. Then, $\mathcal{R}(B_m) = 1/3$ (achieved by the B_m-avoiding set $\{3j : j \ge 0\} \cap [n]$ for all n). Then, according to Proposition 4.4, it is possible to derive that for all $t, s \ge 1$, and $R = \{-2^t, \ldots, -4, -2, -1, 1, 2, 4, \ldots, 2^s\}$, $\mathsf{cap}(\mathbb{Z}_R) \le 1 - 1/3$. Equality holds in this case since by Proposition 4.2, $\mathsf{cap}(\mathbb{Z}_R) \ge \mathsf{cap}(\{-2, -1, 1, 2\}) = 2/3$.

4.2.1. *Interleaving and recoverable systems.* The interleaving construction can also be applied to recoverable systems, and sometimes it yields optimal-rate storage codes on infinite graphs. For instance, let X_R be a one-dimensional recoverable system defined in Proposition 4.2, and let $\mathbb{Z}_R$ be its corresponding (infinite) graph. As in Example 3, we first color $\mathbb{Z}_R$ with $\{0, \ldots, (r+l)\}$ such that vertex $t \in \mathbb{Z}_R$ is colored with $t \bmod (r+l+1)$, and then we find a family of orthogonal partitions $\mathcal{M} = \{M_0, \ldots, M_{l+r}\}$ of shape (k, s). Now define the infinite graph $\overline{\mathbb{Z}}_R$ on the vertices $\mathbb{Z} \times \{1, \ldots, s\}$ such that there is an edge from (t, μ) to (t', μ') if and only if

- there is an edge from t to t' in $\mathbb{Z}_R$, and
- column μ of $M_{t \bmod (r+l+1)}$ intersects column μ' of $M_{t' \bmod (r+l+1)}$.

The interleaved codewords are obtained by placing $(x_t^{\lambda_1}, \dots, x_t^{\lambda_k})^T$ on vertex (t, μ), where $\lambda_i = M_{c(t)}(i, \mu)$, $i = 1, \dots, k$. Letting $x^1, \dots, x^{ks} \in X_R$ run over all the possible choices in X_R, we obtain the interleaved code $\bar{X}_R$. In general, the interleaving construction works for every recoverable system X_R, and yields recoverable systems $\bar{X}_R$ on an infinite graph defined by the interleaving. To define the rate, let Y be a recoverable system defined on $\bar{Z}_R$ and set

$$\mathcal{R}_q(Y) := \lim_{n\to\infty} \frac{1}{ns} \log_q B_{[n]\times\{1,\dots,s\}}(Y),$$

and finally let $\mathsf{cap}_q(\bar{\mathbb{Z}}_R) = \sup_Y \mathcal{R}_q(Y)$. We observe that Lemma 4.1 can be generalized to bound $\mathsf{cap}_{q^k}(\bar{\mathbb{Z}}_R)$, namely, $\mathsf{cap}_{q^k}(\bar{\mathbb{Z}}_R) \le \limsup_{n\to\infty} \mathsf{cap}(\bar{\mathbb{Z}}_R \cap [n] \times \{1, \dots, s\})$ [4]. Therefore, if $\mathsf{cap}_q(X_R)$ is bounded above by the DAG bound (as it happens, for instance, for recoverable systems defined in Propositions 4.2, 4.3, and 4.4), then the interleaving construction yields a capacity-achieving recoverable system $\bar{X}_R$ on $\bar{\mathbb{Z}}_R$.

4.3. **Recoverable systems on $\mathbb{Z}^d$.** In this section, we present results concerning the capacity of multidimensional systems and their relationship with one-dimensional capacity. As in the one-dimensional case, we derive results for the d-dimensional grid relying on capacity estimates for storage codes in its finite subgraphs, arguing that the boundary effects can be disregarded in the limit of large n. Let $\mathbb{Z}^d_{R,n} := \mathbb{Z}^d_R \cap [n]^d$ and observe that Lemma 4.1 affords the following generalization.

Lemma 4.5. *Let $R \subset \mathbb{Z}^d \setminus \{0^d\}$ be a finite set and let $(c_n)_n$ be a sequence of real numbers such that $\mathsf{cap}(\mathbb{Z}^d_{R,n}) \le c_n$ for all $n \ge 1$. Then, $\mathsf{cap}(\mathbb{Z}^d_R) \le \limsup_{n\to\infty} c_n$.*

Proof. Again let $A^d_n := \{m \in [n]^d : m + R \subsetneq [n]^d\}$. Since R is a finite region whose size is independent of n, and since the number of $(d-1)$-dimensional faces of the cube $[n]^d$ is $2d$, we have $|A^d_n| = O_d(n^{d-1})$.

Arguing as in Lemma 4.1, let X be an R-recoverable system, let $z \in \mathcal{Q}^{|A^d_n|}$ be a fixed word, and let $B^z_{[n]^d}$ be the restriction of X to $[n]^d$ that matches z on A^d_n. The set $B^z_{[n]^d}$ forms a storage code for $\mathbb{Z}^d_{R,n}$, and this holds for every choice of z. Since there are $q^{|A^d_n|} = O(q^{n^{d-1}})$ such choices, using the assumption of the lemma, we obtain

$$\frac{1}{n^d} \log_q(O(q^{n^{d-1}})|B^z_{[n]^d}|) \le O\Big(\frac{1}{n}\Big) + c_n.$$

Since this is true for every system X, in the limit of $n \to \infty$ we obtain the claim of the lemma. □

This lemma is used in Section 5 below to derive some capacity results for $\mathbb{Z}^2$. In this section we show that in some cases capacity of graphs over $\mathbb{Z}^d$ is tightly related to capacity of one-dimensional graphs. For the remainder of this section, we assume an additional property, which we call invariance. Recall that if R is a recovery region for $\mathbb{Z}^d$, then it defines a modified graph $\mathbb{Z}^d_R$ obtained by adding edges from every vertex $v \in \mathbb{Z}^d$ to the vertices in its recovery region. We say that R is invariant if for any DAG S in $\mathbb{Z}^d_R$, the shifted set $a + S := \{a + s, s \in S\}$ also forms a DAG set. By extension, an R-recoverable system X on $\mathbb{Z}^d$ is called invariant if so is R. While in general the recovery function depends on the vertex to be recovered, in shift invariant systems, it does not depend on the vertex, and has a fixed shape up to translation.

[4] The proof follows the proof of Lemma 4.1, with the only difference that we take $A_n = \{(m, \mu) \in [n] \times \{1, \dots, s\} : m + R \notin [n]\}$.

We begin by demonstrating that there is no loss of generality in limiting oneself to n-cubes in the capacity definition, Definition 4.3. Consider an R-recoverable system X with a finite recovery region $R \subseteq \mathbb{Z}^d \setminus \{0\}, d \geq 2$.

For any set $S \subseteq \mathbb{Z}^d$ denote by $\mathcal{B}_S(X)$ the restriction of the words in X to the set S. Now assume X is an invariant system, let $S_1, S_2 \subset \mathbb{Z}^d$, and notice that $|\mathcal{B}_{S_1 \cup S_2}(X)| \leq |\mathcal{B}_{S_1}(X)| \cdot |\mathcal{B}_{S_2}(X)|$, or

$$\log_q |\mathcal{B}_{S_1 \cup S_2}(X)| \leq \log_q |\mathcal{B}_{S_1}(X)| + \log_q |\mathcal{B}_{S_2}(X)|.$$

Moreover, by the definition of an R-recoverable system, for any finite set $S \subseteq \mathbb{Z}^d$ and any $a \in \mathbb{Z}^d$ we have $|\mathcal{B}_S(X)| = |\mathcal{B}_{a+S}(X)|$. Our arguments will use the Ornstein-Weiss Theorem (see [23] and a detailed proof in [20]), which we cite in the form adjusted to our needs.

Theorem 4.6. [Ornstein-Weiss] *Let f be a function from the set of finite subsets of $\mathbb{Z}^d$ to $\mathbb{R}$, satisfying the following:*

- *f is sub-additive, i.e., for every finite subsets S_1, S_2, $f(S_1 \cup S_2) \leq f(S_1) + f(S_2)$.*
- *f is translation invariant, i.e., for any $a \in \mathbb{Z}^d$ and $S \subseteq \mathbb{Z}^d$, $f(a + S) = f(S)$.*

Then for every sequence $(S_i)_i \subseteq \mathbb{Z}^d$ such that $\lim_{i\to\infty} \frac{|(a+S_i) \triangle S_i|}{|S_i|} = 0$ for every $a \in \mathbb{Z}^d$, the limit $\lim_{i\to\infty} \frac{f(S_i)}{|S_i|}$ exists, is finite, and it does not depend on the choice of the sequence $(S_i)_i$.

A sequence $(S_i)_i$ for which $\lim_{i\to\infty} \frac{|(a+S_i) \triangle S_i|}{|S_i|} = 0$ for every $a \in \mathbb{Z}^d$, is called a Følner sequence, and it can be thought of as a sequence of shapes whose boundary becomes negligible. Theorem 4.6 implies that for invariant systems, for every Følner sequence $(S_i)_i$ we have

$$\mathcal{R}(X) = \lim_{n\to\infty} \frac{\log_q |\mathcal{B}_{[n]^d}(X)|}{n^d} = \lim_{i\to\infty} \frac{\log_q |X_{S_i}|}{|S_i|}.$$

In particular, since the sequence $([n]^d)_{(n\geq 1)}$ is a Følner sequence, the rate $\mathcal{R}(X)$ is well defined. The capacity of a recovery region R (or of the graph $\mathbb{Z}^d_R$) is defined as the largest attainable rate of R-recoverable systems. In this section we limit ourselves to invariant systems, and we denote their capacity by $\mathsf{cap}_r(R)$ (instead of $\mathsf{cap}_r(\mathbb{Z}^d_R)$), where the subscript r refers to invariance. Thus, $\mathsf{cap}_r(R) = \sup \mathcal{R}(X_R)$, where the supremum is taken over all invariant systems in $Q^{\mathbb{Z}^d}$. Notice that all the definitions given so far are valid when restricting ourselves to the set of invariant recoverable systems.

We will now establish a connection between the capacity of invariant recoverable systems in one dimension and their higher-dimensional counterparts. To do so, we introduce the following definition. Let $\boldsymbol{n} = (n_1, \ldots, n_d) \in \mathbb{N}^d$ be a d-dimensional vector. As above, $\mathcal{B}_{[\boldsymbol{n}]}(X)$ denotes the set of words in X that are restricted to the coordinates $[\boldsymbol{n}] = [n_1] \times \cdots \times [n_d]$. Given a sequence of vectors $(\boldsymbol{n}_i)_i$, we say that $\boldsymbol{n}_i \to \infty$ as $i \to \infty$ if the sequence

$$\min \boldsymbol{n}_i := \min\{(n_1)_i, \ldots, (n_d)_i\} \to \infty$$

as $i \to \infty$. It is straightforward to show that if a sequence $(\boldsymbol{n}_i)_i \subset \mathbb{Z}^d$ satisfying $\lim_{i\to\infty} \boldsymbol{n}_i = \infty$, then $\lim_{i\to\infty} \frac{|(a+[\boldsymbol{n}_i]) \triangle [\boldsymbol{n}_i]|}{|[\boldsymbol{n}_i]|} = 0$ for every $a \in \mathbb{Z}^d$, i.e., $(\boldsymbol{n}_i)_i$ is a Følner sequence (here $|[\boldsymbol{n}]| = \prod_{j=1}^n n_j$).

Let

$$R^d = \bigcup_{i=1}^{d} (\{0\}^{i-1} \times R \times \{0\}^{d-i}) \tag{10}$$

(below we call such regions *axial products*; see Thm. 5.4). It is intuitively clear that $\mathsf{cap}_r(R) \leq \mathsf{cap}_r(R^d)$. Below we formally prove this claim together with an upper bound on $\mathsf{cap}_r(R^d)$. We

will use a multivariate generalization of the well-known Fekete lemma, which extends the original claim to subadditive functions on d-dimensional lattices. A general version of this statement appears, for instance, in [15, Lemma 15.11], although apparently it has been known for a long while. We will use a version adjusted to our needs, considering sequences indexed by integer vectors and subadditive along each of the coordinates.

For a vector $\boldsymbol{n} = (n_1, \dots, n_d) \in \mathbb{N}^d$ and $c \in \mathbb{N}$, let $\boldsymbol{n}^{(i,+c)}$ be the vector obtained from $\boldsymbol{n}$ by replacing n_i with $n_i + c$, and let $\boldsymbol{n}^{(i,c)}$ be obtained upon replacing n_i with c.

Theorem 4.7. [8, Th. 1], [9] *Let $\xi : \mathbb{N}^d \to [0, \infty)$ be a sequence that satisfies*

$$\xi(\boldsymbol{n}^{(i,+c_i)}) \le \xi(\boldsymbol{n}) + \xi(\boldsymbol{n}^{(i,c_i)})$$

for all $c_i \in \mathbb{N}, i = 1, \dots, d$. Then, the limit $\lim_{\boldsymbol{n}\to\infty} \frac{\xi(\boldsymbol{n})}{\prod_j n_j}$ exists and is equal to $\inf_{S\subseteq\mathbb{N}^d} \frac{\xi(S)}{|S|}$.

Proposition 4.8. *Let $R \subset \mathbb{Z} \setminus \{0\}$ be a finite set and for every n, let S_n be an invariant DAG set in $\mathbb{Z}_{R,n}$. Then*

$$\mathsf{cap}_r(R) \le \mathsf{cap}_r(R^d) \le \limsup_{n\to\infty}(1 - |S_n|/n).$$

Proof. It will suffice to limit ourselves to considering restrictions of $X_{R,n}$ to cubes, so below $\mathcal{B}_{[\boldsymbol{n}]}$ denotes a cube $[n]^d$. The first inequality is obvious by stacking one-dimensional words from $X_{R,n}$ to form a word in X_{R^d}. Namely, fix any $(i_1, \dots, i_{d-1}) \in [n]^{d-1}$ and place a word of $X_{R,n}$ in the n positions given by varying the last coordinate i_d inside $\mathcal{B}_{[\boldsymbol{n}]}$; repeat for all the choices of $(i_1, \dots, i_{d-1})$. Since every symbol can be recovered from its one-dimensional neighborhood along the varying coordinate, it can also be recovered from its R^d-neighborhood.

Let us prove the second inequality. For $a \in \mathbb{N}$ we have

$$|\mathcal{B}_{[\boldsymbol{n}^{(i,+a)}]}(X_{R^d})| \le |\mathcal{B}_{[\boldsymbol{n}]}(X_{R^d})| \cdot |\mathcal{B}_{[\boldsymbol{n}^{(i,a)}]}(X_{R^d})|,$$

or

$$\log|\mathcal{B}_{[\boldsymbol{n}^{(i,+a)}]}(X_{R^d})| \le \log|\mathcal{B}_{[\boldsymbol{n}]}(X_{R^d})| + \log|\mathcal{B}_{[\boldsymbol{n}^{(i,a)}]}(X_{R^d})|.$$

Thus, the log-volume sequence is subadditive coordinate-wise, so Theorem 4.7 applies, yielding that the limit in (8) exists, i.e.,

$$\mathcal{R}(X_R) = \lim_{n\to\infty} \frac{\log|\mathcal{B}_{[\boldsymbol{n}]}(X_{R^d})|}{n^d}.$$

Moreover, this limit equals the infimum on the choice of the shape, so taking $\boldsymbol{n}_i = (i, 0, 0, \dots, 0)$ for the upper bound on capacity we obtain that

$$\mathcal{R}(X_{R^d}) \le \frac{\log \mathcal{B}_{[\boldsymbol{n}_i]}(X_{R^d})}{i}.$$

Since $\mathcal{B}_{[\boldsymbol{n}_i]}(X_{R^d})$ is the restriction of X to $[\boldsymbol{n}_i]$, and since R^d is constructed as an axial product, we obtain that S_n is a DAG set in $\mathbb{Z}_{R^d,n}$. Thus, $\frac{\log \mathcal{B}_{[\boldsymbol{n}_i]}(X_{R^d})}{i} \le 1 - |S_n|/n$ which implies that

$$\mathsf{cap}_r(R^d) \le \limsup_{n\to\infty}\Big(1 - \frac{|S_n|}{n}\Big).$$

□

As an immediate corollary of this proposition, we obtain the following.

Corollary 4.9. *Let $R \subseteq \mathbb{Z} \setminus \{0\}$ be a finite set and let X be an invariant R-recoverable system that attains the MAIS bound (3). Let R^d be given by (10), then $\mathsf{cap}(R^d) = \mathsf{cap}(R)$.*

5. Capacity of $\mathbb{Z}^2$ with l_1 and l_∞ recovery regions

The results in the previous sections enable us to study the capacity of graphs that represent a storage network over the two-dimensional grid. We start with recovery sets formed by radius-r balls under the l_1 and l_∞ metrics. For $v=(i_1,j_1), u=(i_2,j_2)\in\mathbb{Z}^2$,

$$d_1\big((i_1,j_1),(i_2,j_2)\big)=|i_1-i_2|+|j_1-j_2|,$$
$$d_\infty\big((i_1,j_1),(i_2,j_2)\big)=\max\{|i_1-i_2|,|j_1-j_2|\}.$$

The l_1 metric on $\mathbb{Z}^2$ is sometimes called the Manhattan distance (or even Lee distance [13], although this usage is not fully accurate). A sphere in this metric is a rhombic pattern whose exact shape depends on the parity of the radius (see below). A sphere in the metric d_∞ is a $(2r+1)\times(2r+1)$ square. In both cases to argue about recovery, we add to $\mathbb{Z}^2$ the edges that connect every vertex with its neighbors in the sphere of radius r about it, and denote the resulting graph by G_r^α, for $\alpha=1,\infty$. To find the capacity of these graphs, we rely on Theorem 2.2 applied to their finite subgraphs. These subgraphs do not have transitive automorphisms, but finding the capacity of the system essentially amounts to constructing a perfect covering of the graph by anticodes. This claim is proved in the following general statement.

Proposition 5.1. *Let $D_G(r)$ be the largest size of an anticode of diameter r in $\mathbb{Z}^2$ in the d_α metric, $\alpha=1,\infty$, and suppose that $\mathbb{Z}^2$ admits a tiling with anticodes of size $D_G(r)$. Then*

$$\mathsf{cap}(G_r^\alpha)=1-\frac{1}{D_G(r)}.$$

Proof. Suppose that there is a perfect covering of $\mathbb{Z}^2$ with largest-size anticodes of diameter r. Then the graph $G:=\mathbb{Z}_n^2$ contains a perfect covering, except for the subset formed of the points at distance r or less from one of the boundaries. As a result, $\mathsf{cap}(G_r)\le 1-\frac{1}{D_G(r)}$, and also $\mathsf{cap}((\mathbb{Z}^2)_r)\le 1-\frac{1}{D_G(r)}$ on account of Lemma 4.5. At the same time, since the anticodes provide a perfect covering of $\mathbb{Z}^2$, we have that $\mathsf{cap}(G_r)\ge 1-\frac{1}{D_G(r)}$. □

This enables us to find the capacity values of the graphs G_r^α.

Theorem 5.2. *For all $r\ge 1$ it holds that*

$$\mathsf{cap}(G_r^1)=1-\frac{1}{D_1(r)},$$
$$\mathsf{cap}(G_r^\infty)=1-\frac{1}{(r+1)^2},$$

where $D_1(r)$ is the size of a maximum anticode of diameter r in the l_1 sphere, and

$$D_1(r)=\begin{cases}\frac{(r+1)^2}{2} & r \text{ odd},\\ \frac{r^2}{2}+r+1 & r \text{ even}.\end{cases}$$

Proof. For the graph G_r^∞, anticodes of diameter r are simply squares of size $(r+1)\times(r+1)$, so $D_{G_r^\infty}(r)=(r+1)^2$. The squares tile the graph, giving a perfect covering, and thus

$$\mathsf{cap}(G_r^\infty)=1-\frac{1}{(r+1)^2}.$$

The result for the G_r^1 metric is obtained from perfect tiling which exists for maximal anticodes in the l_1 metric (see e.g. [13]). In detail, for $r=2k$, the largest anticode with diameter r is an l_1 ball of radius k, which forms a tiling in $\mathbb{Z}^2$. Its size is easily found by induction to be $r^2/2+r+1$. For $r=2k+1$, the largest anticode $\mathscr{D}_k$ can be defined recursively as follows. Let $\mathscr{D}_0$ be a shape

formed by two adjacent (with l_1 distance one) points of $\mathbb{Z}^2$, and let $\mathscr{D}_m, m = 1, \ldots, k$ be formed of $\mathscr{D}_{m-1}$ and all the points that are adjacent to at least one of points in $\mathscr{D}_{m-1}$. Note that $\mathscr{D}_k$ also generates a tiling, and $|\mathscr{D}_k| = (r+1)^2/2$. □

Remark 4. Note that this theorem relies on a stronger assumption that Theorem 2.2, namely that the there is a tiling of the graph with translations of a largest-size anticode. The existence of tilings and more generally, diameter perfect codes in $\mathbb{Z}^d$, is an active research topic, see [30, 31] for recent additions to the literature.

Now suppose that the recovery set is not symmetric, for instance, a direct product of two non-symmetric segments. We have the following result.

Theorem 5.3. *Let l, r, b, a be positive integers and consider the recovery set $R = [-l, r] \times [-b, a]$ where $0 \le r < l$ and $0 \le b < a$. Let X_R be a two-dimensional R-recoverable system on $\mathbb{Z}^2$. Then*

$$\mathsf{cap}(G_R) = 1 - \frac{1}{(r+1)(b+1)}. \tag{11}$$

Proof. To show that $\mathsf{cap}(G_R)$ is less than the right-hand side of (11) we observe that

$$S = \{(i(r+1), j(b+1)) : i, j \ge 0\}$$

is a DAG set in the graph G_R. Indeed, there is an edge from $v_1 = (i_1(r+1), j_1(b+1))$ to $v_2 = (i_2(r+1), j_2(b+1))$ only if $i_1 \le i_2$ and $j_1 \le j_2$. Through this, we can define a topological order of all vertices in S and the claim follows from Theorem 2.1.

For the lower bound, let

$$S_1 = \left\{(i, j) \in \mathbb{Z}^2 \ : \ i \in [r], j \in -[b]\right\}.$$

For $x, y \ge 0$, the subset of nodes $(x(r+1), y(b+1)) + S_1$ forms a clique. Since $|S| = |S_1|$, the proof is concluded by (1). □

As already noted (10), one way to construct two-dimensional systems is to form an axial product of two one-dimensional systems. As before, for positive integers l, r, b, a let $R_1 = [-l, r]$ and $R_2 = [-b, a]$. Fix a finite alphabet $\mathcal{Q}$ and consider one-dimensional recoverable systems $Y_1 = Y_{R_1}$ and $Y_2 = Y_{R_2}$. The *axial product* of Y_1 and Y_2 is a two-dimensional system $X = Y_1 \times Y_2$ over $\mathcal{Q}$ with the recovery set $(R_1 \times \{0\}) \cup (\{0\} \times R_2)$. In words, in the system X the symbol x_i is a function of the l symbols to its left, r symbols to its right, a symbols above, and b symbols below it. The axial product construction is different from the direct product $[-l, r] \times [-b, a]$ in that it results in a cross-shaped rather than a rectangular recovery region.

Theorem 5.4. *For given positive integers l, r, b, a, let $R_1 = [-l, r]$ and $R_2 = [-b, a]$, and let $X = Y_1 \times Y_2$ be the axial product of the one-dimensional systems $Y_1 = Y_{R_1}$ and $Y_2 = Y_{R_2}$. Denote the recovery set $(R_1 \times \{0\}) \cup (\{0\} \times R_2)$ by R^2. Then the capacity is*

$$\mathsf{cap}(\mathbb{Z}^2_{R^2}) = \frac{t}{t+1},$$

where $t = \max\{\min\{l, r\}, \min\{a, b\}\}$. Moreover, a system X that attains this value can be obtained from the one dimensional R-recoverable system with $R = \{j : 0 < |j| < t\}$.

Proof. We first observe that $\mathsf{cap}(\mathbb{Z}^2_{R^2}) \ge \max\{\mathsf{cap}(\mathbb{Z}_{R_1}), \mathsf{cap}(\mathbb{Z}_{R_2})\}$ by stacking words from Y_{R_1} or Y_{R_2} on top of each other to generate a recoverable system. The recovery set is $\{R_1 \times \{0\}\} \subset R^2$ or $\{\{0\} \times R_2\} \subset R^2$. Then by Proposition 4.2 we have

$$\mathsf{cap}(\mathbb{Z}^2_{R^2}) \ge \max\left\{\lim_{n\to\infty} \frac{|\mathbb{Z}_{R_1}|n}{n^2}, \lim_{n\to\infty} \frac{|\mathbb{Z}_{R_2}|n}{n^2}\right\} = \max\{\mathsf{cap}(\mathbb{Z}_{R_1}), \mathsf{cap}(\mathbb{Z}_{R_2})\} = \frac{t}{t+1}.$$

To show the other direction, note that the set of vertices

$$S := [n]^2 \cap \{(k(t+1),0)+(i,i),(0,k(t+1))+(i,i), k,i \in \mathbb{Z}\}$$

is a DAG set in $\mathbb{Z}^2_{R^2,n}$ (as before, $\mathbb{Z}^2_{R^2,n} := \mathbb{Z}^2_{R^2} \cap [n]^2$). It remains to calculate the cardinality of S. Assume without loss of generality that $n = m(t+1)$ for some $m \in \mathbb{N}$, then we have

$$\begin{aligned} |S| &= 2(1+(t+1)+2(t+1)+\cdots+m(t+1))-n \\ &= 2+nm \\ &= 2+\frac{n^2}{t+1}. \end{aligned}$$

Thus,

$$\mathsf{cap}(\mathbb{Z}^2_{R^2}) \leq \limsup_{n\to\infty}\left(1-\frac{|S|}{n^2}\right) = \frac{t}{t+1}.$$

□

This theorem shows that for some recoverable systems, the two-dimensional capacity is not increased from the one-dimensional one: the maximally sized system is obtained by stacking independent one-dimensional systems in the rows (or the columns). While the one-dimensional components in the axial product provide an obvious lower bound on the capacity of the two-dimensional system, it is not clear whether the equality holds in all cases. We could not find examples of axial products with higher capacity by relying on both dimensions, and we leave this question as an open problem.

6. Conclusion

In this paper, we studied storage codes on finite graphs and their extensions to $\mathbb{Z}$ and $\mathbb{Z}^2$, which we call recoverable systems. Earlier results for recoverable systems [12] relied on the assumption of shift invariance. Lifting this restriction enables one to connect it to the line of research on storage codes for finite graphs. Relying on resolvable designs, we proposed a new way of propagating storage codes by interleaving, which yields many new families from the known ones. Using bounds and constructions for finite storage codes, we found capacity of several recovery sets, such as l_1 or l_∞ balls or cross-shaped regions. Finally, we established a link of the capacity problem to questions in additive combinatorics related to difference-avoiding sets, and found capacity values for some special recovery sets.

There are numerous open problems that we leave for future research. To point out some of them, one may ask what is the maximum capacity of a two-dimensional system when only a subset of neighbors can be used for recovery. Next, it appears that the capacity of an axial product is always attained by stacking one-dimensional systems, although proving this has been elusive. If not true, then what are the conditions under which this is the case? Among other questions: How does the fact that the order of the neighbors is known affect the capacity of the system? And finally, is it possible to characterize the number of recovery functions needed to obtain the maximum capacity?

Appendix A. Linear programming bound

To describe the linear programming bound of Mazumdar et al. [22], let us first define a τ-cover by gadgets on a graph $G = (V,E)$. For a subset of vertices $A \subset G$, let $\mathrm{cl}(A) := \{v : \mathcal{N}(v) \subseteq A\}$ be the closure of A which contains all vertices in A and their neighbors. A tuple $g = (S_1, S_2, c_1, c_2)$ is called a *gadget* if

(1) There exist two sets of vertices $A, B \subseteq V$ such that $S_1 = \mathrm{cl}(A) \cup \mathrm{cl}(B)$ and $S_2 = \mathrm{cl}(A) \cap \mathrm{cl}(B)$. We call S_1 the outside and S_2 the inside of the gadget.
(2) Colors c_1 and c_2 are picked from a fixed set of size τ, and they are assigned to all the vertices in S_1, S_2 respectively. Note that each vertex can have multiple assigned colors.

We call $(\{v\}, \emptyset, c_1, c_2)$ a trivial gadget, and $w(g) = |A|+|B|$ the weight of gadget g. A collection of gadgets forms a τ-cover if, for every color c, all the vertices with color c form a vertex cover. It was shown in [22, Theorem 8] that the total weight of the gadgets that form a τ-cover provides an upper bound on $\mathsf{cap}(G)$. More formally, we have the following theorem.

Theorem A.1 (Linear programming (LP) bound [22])**.** *Let $\tau > 0$ be a fixed integer. For each gadget that contains $S \subseteq V$, define $\chi_{g,S} = \mathbb{1}\{g \text{ is involved in the cover}\}$, where S is a part of g. Thus, each gadget g corresponds to two variables χ_{g,S_1} and χ_{g,S_2}, where S_1 and S_2 are the outside and the inside of g, respectively. Denote by $c_g(S)$ the color of the vertex set S in gadget g.*

The capacity $\mathsf{cap}(G)$ is bounded above by the solution to the following linear program:

$$\textit{minimize} \quad \frac{1}{n\tau} \sum_{g,S} \chi_{g,S} \frac{w(g)}{2}$$

$$\textit{s.t.,} \quad \frac{1}{\tau} \sum_{g,S} \chi_{g,S} \frac{w(g)}{2} \in \mathbb{N}$$

$$\sum_{\substack{g,S: u \in S \\ C_g(S)=c}} \chi_{g,S} + \sum_{\substack{g',S': v \in S' \\ C_{g'}(S')=c}} \chi_{g',S'} \geq 1, \quad \forall (u,v) \in E, \forall c \tag{12}$$

$$\chi_{g,S_1} = \chi_{g,S_2}, \quad \forall g \text{ with outside } S_1 \text{ and inside } S_2. \tag{13}$$

Note that conditions (12) guarantee that the sets form a vertex cover, and (13) states that the inside is involved in the gadget cover if and only if the outside is also involved.

Proof of Proposition 3.7: Let $g = (S_1, S_2, c_1, c_2)$ be a gadget on G and define $\bar{g} = (\bar{S}_1, \bar{S}_2, c_1, c_2)$, where $S_1 := \bigcup_{v \in S_1} \bar{v}$ and $S_2 := \bigcup_{v \in S_2} \bar{v}$. If $g_1, \ldots, g_m$ forms a τ-cover of G, then it is straightforward to check that $\bar{g}_1, \ldots, \bar{g}_m$ forms a τ-cover of $\bar{G}$. Note that $w(\bar{g}) = sw(g)$, thus $\frac{1}{n\tau} w(g) = \frac{1}{ns\tau} w(\bar{g})$, and $\bar{g}_1, \ldots, \bar{g}_m$ induce an upper bound that equals to the bound by $g_1, \ldots, g_m$, which is $\mathsf{cap}(G)$ by our assumption. In other words, we have $\mathsf{cap}_{q^k}(\bar{G}) \leq \mathsf{cap}(G)$. The matching lower bound follows by Proposition 3.3.

CONFLICT OF INTERESTS STATEMENT

The authors hereby declare that they have no known conflicts of interest or competing interests related to the research reported in this manuscript.

DATA AVAILABILITY STATEMENT

This paper has no associated data.

REFERENCES

[1] R. Ahlswede, H. Aydinian, and L. Khachatryan, "On perfect codes and related concepts," *Des. Codes Cryptogr.*, vol. 22, no. 3, pp. 221–237, 2001.

[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 *53rd Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, 2015, pp. 843–850.

[3] ——, "Fundamentals of index coding," *Found. Trends Commun. Inf. Theory*, vol. 14, no. 3-4, pp. 163–346, 2018.

[4] Z. Bar-Yossef, Y. Birk, T. S. Jayram, and T. Kol, "Index coding with side information," *IEEE Trans. Inf. Theory*, vol. 57, no. 3, pp. 1479–1494, 2011.

[5] A. Barg, O. Elishco, R. Gabrys, and E. Yaakobi, "Recoverable systems on lines and grids," in *2022 IEEE Int. Symp. Inf. Theory*, 2022, pp. 2637–2642.

[6] A. Barg, M. Schwartz, and L. Yohananov, "Storage codes on coset graphs with asymptotically unit rate," 2023, arxiv:2212.1217v2.

[7] A. Barg and G. Zémor, "High-rate storage codes on triangle-free graphs," *IEEE Trans. Inf. Theory*, vol. 68, no. 12, pp. 7787–7797, 2022.

[8] S. Capobianco, "Multidimensional cellular automata and generalization of Fekete's lemma," *Discrete Math. Theor. Comput. Sci.*, vol. 10, no. 3, pp. 95–104, 2008.

[9] ——, "Fekete's lemma for componentwise subadditive functions of two or more real variables," *Acta Comment. Univ. Tartu. Math.*, vol. 26, no. 1, pp. 45–62, 2022.

[10] C. J. Colbourn and J. H. Dinitz, Eds., *Handbook of combinatorial designs*, 2nd ed., ser. Discrete Mathematics and its Applications (Boca Raton). Chapman & Hall/CRC, Boca Raton, FL, 2007.

[11] P. Delsarte, "An algebraic approach to the association schemes of coding theory," *Philips Res. Repts. Suppl.*, no. 10, 1973.

[12] O. Elishco and A. Barg, "Recoverable systems," *IEEE Trans. Inf. Theory*, vol. 68, no. 6, pp. 3681–3699, 2022.

[13] T. Etzion, "Product constructions for perfect Lee codes," *IEEE Trans. Inf. Theory*, vol. 57, no. 11, pp. 7473–7481, 2011.

[14] ——, *Perfect Codes and Related Structures*. World Scientific, 2022.

[15] H.-O. Georgii, *Gibbs measures and phase transitions*, 2nd ed. Berlin/New York: Walter de Gruyter & Co., 2011.

[16] A. Golovnev and I. Haviv, "The (generalized) orthogonality dimension of (generalized) Kneser graphs: Bounds and applications," *Theory of Computing*, vol. 18, pp. 1–22, 2022, also: Proceedings of the 36th Computational-Complexity Conference, 2021.

[17] 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, 2011.

[18] H. Huang and Q. Xiang, "Construction of storage codes of rate approaching one on triangle-free graphs," *Des. Codes Cryptogr.*, vol. 91, no. 12, pp. 3901–3913, 2023.

[19] D. E. Knuth, *The Art of Computer Programming. Vol. 1: Fundamental Algorithms*, 2nd ed. Reading, MA: Addison-Wesley Pub. Co., 1973.

[20] F. Krieger, "The Ornstein-Weiss lemma for discrete amenable groups," Max Planck Institute for Mathematics, MPIM Preprint 2010-48, 2010.

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

[22] 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, 2019.

[23] D. S. Ornstein and B. Weiss, "Entropy and isomorphism theorems for actions of amenable groups," *J. Analyse Math.*, vol. 48, pp. 1–141, 1987.

[24] V. Ramkumar, S. B. Balaji, B. Sasidharan, M. Vajha, M. N. Krishnan, and P. V. Kumar, "Codes for distributed storage," *Found. Trends Commun. Inf. Theory*, vol. 19, no. 4, pp. 547–813, 2022.

[25] A. Rice, "A maximal extension of the best-known bounds for the Furstenberg-Sárközy theorem," *Acta Arith.*, vol. 187, no. 1, pp. 1–41, 2019.

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

[27] A. Sárkőzy, "On difference sets of sequences of integers. I," *Acta Math. Acad. Sci. Hungar.*, vol. 31, no. 1-2, pp. 125–149, 1978.

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

[29] M. Shi, Y. Xia, and D. S. Krotov, "A family of diameter perfect constant-weight codes from Steiner systems," *J. Combin. Theory Ser. A*, vol. 200, pp. Paper No. 105 790, 20, 2023.

[30] T. Zhang and G. Ge, "On linear diameter perfect Lee codes with diameter 6," *J. Combin. Theory Ser. A*, vol. 201, 2024, paper No. 105816.
[31] T. Zhang and Y. Zhou, "On the nonexistence of lattice tilings of Z^n by Lee spheres," *J. Combin. Theory Ser. A*, vol. 165, pp. 225–257, 2019.