Time–Space Constrained Codes for Phase-Change Memories

Minghai Qin, *Student Member, IEEE*, Eitan Yaakobi, *Member, IEEE*, and Paul H. Siegel, *Fellow, IEEE*

***Abstract*—Phase-change memory (PCM) is a promising nonvolatile solid-state memory technology. A PCM cell stores data by using its amorphous and crystalline states. The cell changes between these two states using high temperature. However, since the cells are sensitive to high temperature, it is important, when programming cells (i.e., changing cell levels), to balance the heat both in time and in space. In this paper, we study the time–space constraint for PCM, which was originally proposed by Jiang and coworkers. A code is called an (α, β, p)-*constrained code* if for any α consecutive rewrites and for any segment of β contiguous cells, the total rewrite cost of the β cells over those α rewrites is at most p. Here, the cells are binary and the rewrite cost is defined to be the Hamming distance between the current and next memory states. First, we show a general upper bound on the achievable rate of these codes which extends the results of Jiang and coworkers. Then, we generalize their construction for $(\alpha \geqslant 1, \beta = 1, p = 1)$-constrained codes and show another construction for $(\alpha = 1, \beta \geqslant 1, p \geqslant 1)$-constrained codes. Finally, we show that these two constructions can be used to construct codes for all values of α, β, and p.**

***Index Terms*—Constrained codes, phase-change memory, write-once memory codes.**

I. Introduction

PHASE-CHANGE memory (PCM) devices are a promising technology for nonvolatile memories. Like a flash memory, a PCM consists of cells that can be in distinct physical states. In the simplest case, the PCM cell has two possible states: an amorphous state and a crystalline state. Multiple-bit per cell PCMs can be implemented by using partially crystalline states [4].

Whereas in a flash memory one can decrease a cell level only by erasing the entire block of about 10^6 cells that contains it, in a PCM one can independently decrease an individual cell level—but only to level zero. This operation is called a RESET operation. A SET operation can then be used to change the cell state to any valid level. Therefore, in order to decrease a cell level from one nonzero value to a smaller nonzero value, one needs to first RESET the cell to level zero, and then SET it to the new desired level [4]. Thus, as with flash memory programming, there is a significant asymmetry between the two operations of increasing and decreasing a cell level.

As in a flash memory, a PCM cell has a limited lifetime; the cells can tolerate only about $10^7 - 10^8$ RESET operations before beginning to degrade [12]. Therefore, it is still important when programming cells to minimize the number of RESET operations. Furthermore, a RESET operation can negatively affect the performance of a PCM in other ways. One of them is due to the phenomenon of thermal crosstalk. When a cell is RESET, the levels of its adjacent cells may inadvertently be increased due to heat diffusion associated with the operation [4], [23]. Another problem, called thermal accumulation, arises when a small area is subjected to a large number of program operations over a short period of time [4], [23]. The resulting accumulation of heat can significantly limit the minimum write latency of a PCM, since the programming accuracy is sensitive to temperature. It is therefore desirable to balance the thermal accumulation over a local area of PCM cells in a fixed period of time. Coding schemes can help overcome the performance degradation resulting from these physical phenomena. Lastras-Montaño *et al.* [19] studied the capacity of a write-efficient memory [1] for a cost function that is associated with the write model of PCMs described earlier.

Jiang *et al.* [17] have proposed codes to mitigate thermal crosstalk and heat accumulation effects in PCM. Under their thermal crosstalk model, when a cell is RESET, the levels of its immediately adjacent cells may also be increased. Hence, if these neighboring cells exceed their target level, they also will have to be RESET, and this effect can then propagate to many more cells. In [17], they considered a special case of this and proposed the use of constrained codes to limit the propagation effect. Capacity calculations for these codes were also presented.

The other problem addressed in [17] is that of heat accumulation. In this model, the *rewrite cost* is defined to be the number of programmed cells, i.e., the Hamming distance between the current and next cell-state vectors. A code is said to be

Manuscript received July 18, 2012; revised February 08, 2013; accepted February 22, 2013. Date of publication April 12, 2013; date of current version July 10, 2013. This work was done while E. Yaakobi was with the Department of Electrical and Computer Engineering, University of California, San Diego. This work was supported in part by the International Sephardic Education Foundation, the Lester Deutsch Fellowship, the University of California Lab Fees Research Program under Award 09-LR-06-118620-SIEP, the National Science Foundation under Grant CCF-1116739, and the Center for Magnetic Recording Research at the University of California, San Diego. This paper was presented in part at the 2011 IEEE Global Telecommunications Conference.

M. Qin and P. H. Siegel are with the Department of Electrical and Computer Engineering and the Center for Magnetic Recording Research, University of California, San Diego, La Jolla, CA 92093 USA (e-mail: mqin@ucsd.edu; psiegel@ucsd.edu).

E. Yaakobi is with the Department of Electrical Engineering, California Institute of Technology, Pasadena, CA 91125 USA, and also with the Center for Magnetic Recording Research, University of California, San Diego, La Jolla, CA 92093 USA (e-mail: yaakobi@caltech.edu).

Communicated by N. Kashyap, Associate Editor for Coding Theory.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIT.2013.2257916

(α, β, p)-*constrained* if for any α consecutive rewrites and for any segment of β contiguous cells, the total rewrite cost of the β cells over those α rewrites is at most p. A specific code construction was given for the $(\alpha \geqslant 1, \beta = 1, p = 1)$-constraint as well as an upper bound on the achievable rate of codes for this constraint. An upper bound on the achievable rate was also given for $(\alpha = 1, \beta \geqslant 1, p = 1)$-constrained codes.

The work in [17] dealt with only a few instances of the parameters α, β, and p. In this paper, we extend the code constructions and achievable-rate bounds to a larger portion of the parameter space. In Section II, we formally define the constrained-coding problem for PCM. In Section III, using connections to 2-D constrained coding, we present a scheme to calculate an upper bound on the achievable rate for all values of α, β, and p. If the value of α or β is 1, then the 2-D constraint becomes a 1-D constraint and we calculate the upper bound on the achievable rate for all values of p. This result coincides with the result in [17] for $(\alpha \geqslant 1, \beta = 1, p = 1)$ and $(\alpha = 1, \beta \geqslant 1, p = 1)$. We also derive upper bounds for some cases with parameters satisfying $(\alpha > 1, \beta > 1, p = 1)$ using known results on the upper bound of 2-D constrained codes. In Section IV, several code constructions are given. First, we describe an elementary construction for arbitrary values of α, β, and p. We then show an improved construction for $(\alpha = 1, \beta \geqslant 1, p \geqslant 1)$-constrained codes and extend the construction in [17] of $(\alpha \geqslant 1, \beta = 1, p = 1)$-constrained codes to arbitrary p. Finally, we show how to extend the improved constructions to arbitrary values of α, β, and p.

II. Preliminaries

In this section, we give a formal definition of the constrained-coding problem. The number of cells is denoted by n and the memory cells are binary. The cell-state vectors are the binary vectors from $\{0,1\}^n$. If a cell-state vector $\boldsymbol{u} = (u_1, \ldots, u_n) \in \{0,1\}^n$ is rewritten to another cell-state vector $\boldsymbol{v} = (v_1, \ldots, v_n) \in \{0,1\}^n$, then the rewrite cost is defined to be the Hamming distance between $\boldsymbol{u}$ and $\boldsymbol{v}$, that is,

$$d_H(\boldsymbol{u}, \boldsymbol{v}) = |\{i \,:\, u_i \neq v_i, 1 \leqslant i \leqslant n\}|.$$

The Hamming weight of a vector $\boldsymbol{u}$ is $wt(\boldsymbol{u}) = d_H(\boldsymbol{u}, \boldsymbol{0})$. The complement of a vector $\boldsymbol{u}$ is $\overline{\boldsymbol{u}} = (\overline{u}_1, \ldots, \overline{u}_n)$. For a vector $\boldsymbol{x} = (x_1, \ldots, x_n)$, we denote by $\boldsymbol{x}_p^q$ the subvector $(x_p, x_{p+1}, \ldots, x_q)$ and for a sequence of vectors $\boldsymbol{x}_i = (x_{i,1}, \ldots, x_{i,n})$, $i \in \mathbb{N}$, we denote by $\boldsymbol{x}_{i,p}^q$ the subvector $(x_{i,p}, x_{i,p+1}, \ldots, x_{i,q})$, for $1 \leqslant p \leqslant q \leqslant n$. The set $\{i, i+1, \ldots, j\}$ is denoted by $[i : j]$ for $i \leqslant j$, and in particular, $\{1, 2, \ldots, \lfloor 2^{nR} \rfloor\}$ is denoted by $[1 : 2^{nR}]$ for an integer n and real R.

We will specify a code by an explicit construction of its encoding and decoding maps. On the ith write, for $i \geqslant 1$, the encoder

$$\mathcal{E}_i : [1 : 2^{nR_i}] \times \{0,1\}^n \mapsto \{0,1\}^n$$

maps the new information symbol and the current cell-state vector to the next cell-state vector. The decoder

$$\mathcal{D}_i : \{0,1\}^n \mapsto [1 : 2^{nR_i}]$$

maps the cell-state vector to the represented information symbol. We denote the *individual rate* on the ith write of a code by R_i. Note that the alphabet size of the messages on each write does not have to be the same. The *rate* R of a code is defined as

$$R = \liminf_{m \to \infty} \frac{\sum_{i=1}^{m} R_i}{m}. \tag{1}$$

Remark 1: The limit R exists: since the individual rates R_i, $i \in \mathbb{N}$, are bounded from above by 1, so are the average rates $\frac{\sum_{i=1}^{m} R_i}{m}$, for all $m \geqslant 1$.

Definition 1: Let α, β, p be positive integers. A code $\mathcal{C}$ satisfies the (α, β, p) *time–space constraint* (or simply (α, β, p)-constraint) if for any α consecutive rewrites and for any segment of β contiguous positions, the total rewrite cost of those β positions over those α rewrites is at most p. That is, if $\boldsymbol{v}_i = (v_{i,1}, \ldots, v_{i,n})$, for $i \geqslant 1$, is the cell-state vector on the ith write, then for all $i \geqslant 1$ and $1 \leqslant j \leqslant n - \beta + 1$

$$\left| \{(k, \ell) : v_{i+k, j+\ell} \neq v_{i+k+1, j+\ell}, 0 \leqslant k < \alpha, 0 \leqslant \ell < \beta\} \right| \leqslant p$$

or equivalently

$$\sum_{k=0}^{\alpha-1} d_H(\boldsymbol{v}_{i+k,j}^{j+\beta-1}, \boldsymbol{v}_{i+k+1,j}^{j+\beta-1}) \leqslant p.$$

We call such a code $\mathcal{C}$ an (α, β, p)-*constrained code*.

We assume that the number of writes is large and in the constructions we present there will be a periodic sequence of writes. Thus, it will be possible to change any (α, β, p)-constrained code $\mathcal{C}$ with varying individual rates to an (α, β, p)-constrained code $\mathcal{C}'$ with fixed individual rates such that the rates of the two constrained codes are the same. This can be achieved by using multiple copies of the code $\mathcal{C}$ and in each copy of $\mathcal{C}$ to start writing from a different write within the period of writes. Therefore, we assume that there is no distinction between the two cases and the rate is as defined in (1), which is the average number of bits written per cell per write.

The encoding and decoding maps can be either the same on all writes or can vary among the writes. In the latter case, we will need more cells in order to record the index of the write number. However, arguing as in [28] and [29], it is possible to show that these extra cells do not reduce the asymptotic rate and therefore we assume here that the encoder and decoder know the write number.

A rate R is called an (α, β, p)-*achievable rate* if there exists a sequence of (α, β, p)-constrained codes of increasing length n such that the rate of each code is R. The (α, β, p)-*capacity* of the (α, β, p)-constraint is denoted by $C(\alpha, \beta, p)$ and is defined to be

$$C(\alpha, \beta, p) = \sup R$$

where R is an (α, β, p)-achievable rate.

Our goal in this paper is to give lower and upper bounds on the (α, β, p)-capacity, $C(\alpha, \beta, p)$, for all values of α, β, and p. Clearly, if $p \geqslant \alpha\beta$, then $C(\alpha, \beta, p) = 1$. So we assume throughout the paper that $p < \alpha\beta$. Lower bounds will be inferred from specific constrained code constructions, while the

upper bounds will be derived analytically using tools drawn from the theory of 1-D and 2-D constrained codes.

III. Upper Bound on the Capacity

In this section, we will present upper bounds on the (α, β, p)-capacity obtained using techniques from the analysis of 2-D constrained codes. There are a number of 2-D constraints that have been extensively studied, e.g., 2-D (d, k)-runlength-limited (RLL) constraints [18], [25], the no isolated bits (n.i.b) constraint [11], [13], and the family of checkerboard constraints [22], [27]. Given a 2-D constraint S, its capacity is defined to be

$$C_{2D}(S) = \lim_{m,n\to\infty} \frac{\log_2 c_S(m,n)}{mn}$$

where $c_S(m,n)$ is the number of $m \times n$ arrays that satisfy the constraint S. The constraint of interest for us in this study is the one where in each rectangle of size $a \times b$, the number of ones is at most p.

Definition 2: Let a, b, p be positive integers. An $(m \times n)$-array $A = (a_{i,j})_{1\leqslant i\leqslant m, 1\leqslant j\leqslant n} \in \{0,1\}^{m\times n}$ is called an (a, b, p)-*array* if in each subarray of A of size $a \times b$, the number of 1's is at most p. That is, for all $1 \leqslant i \leqslant m-a+1$, $1 \leqslant j \leqslant n-b+1$

$$\left|\{(k,\ell) : 0 \leqslant k \leqslant a-1, 0 \leqslant \ell \leqslant b-1, a_{i+k,j+\ell} = 1\}\right| \leqslant p.$$

The capacity of the constraint is denoted by $C_{2D}(a, b, p)$.

Note that when $p = 1$, the $(a, a, 1)$-constraint coincides with the square checkerboard constraint of order $a - 1$ [27].

The connection between the capacity of the 2-D constraint $C_{2D}(a, b, p)$ and the (α, β, p)-capacity is the following.

Theorem 1: For all α, β, p, $C(\alpha, \beta, p) \leqslant C_{2D}(\alpha, \beta, p)$.

Proof: Let $\mathcal{C}$ be an (α, β, p)-constrained code of length n. For any sequence of m writes, let us denote by $\boldsymbol{v}_i$, for $i \geqslant 0$, the cell-state vector on the ith write, where $\boldsymbol{v}_0$ is the all-zero vector. The $(m \times n)$-array $A = (a_{i,j})$ is defined to be

$$a_{i,j} = v_{i,j} + v_{i-1,j}$$

where the addition is a modulo-2 sum. That is, $a_{i,j} = 1$ if and only if the jth cell is changed on the ith write. Since $\mathcal{C}$ is an (α, β, p)-constrained code, for all $1 \leqslant i \leqslant m-\alpha$ and $1 \leqslant j \leqslant n-\beta+1$

$$\left|\{(k,\ell) : v_{i+k,j+\ell} \neq v_{i+k+1,j+\ell}, 0 \leqslant k < \alpha, 0 \leqslant \ell < \beta\}\right| \leqslant p$$

and therefore

$$\left|\{(k,\ell) : 0 \leqslant k \leqslant \alpha-1, 0 \leqslant \ell \leqslant \beta-1, a_{i+k,j+\ell} = 1\}\right| \leqslant p.$$

Thus, A is an (α, β, p)-array of size $m \times n$.

Every write sequence of the code $\mathcal{C}$ corresponds to an (α, β, p)-array, and thus, the number of write sequences of length m is at most the number of (α, β, p)-arrays, which is upper bounded by $2^{mnC_{2D}(\alpha,\beta,p)}$, for m, n large enough. Hence, the number of distinct write sequences is at most $2^{mnC_{2D}(\alpha,\beta,p)}$. However, if the individual rate on the ith write is R_i, then the total number of distinct write sequences is $\prod_{i=1}^{m} 2^{nR_i}$. We conclude that

$$\prod_{i=1}^{m} 2^{nR_i} \leqslant 2^{mnC_{2D}(\alpha,\beta,p)}$$

and therefore

$$\frac{\sum_{i=1}^{m} R_i}{m} \leqslant C_{2D}(\alpha, \beta, p).$$

If m goes to infinity, the rate of any (α, β, p)-constrained code R satisfies

$$R \leqslant C_{2D}(\alpha, \beta, p)$$

i.e., $C(\alpha, \beta, p) \leqslant C_{2D}(\alpha, \beta, p)$. ■

Theorem 1 provides a scheme to calculate an upper bound on the (α, β, p)-capacity from an upper bound on the capacity of a 2-D constraint. Unfortunately, good upper bounds are known only for some special cases of the values of α, β, p, and in particular, when $p = 1$. More generally, finding the capacity of 2-D constrained systems, such as those mentioned previously, is a difficult open problem that has attracted considerable attention over the past 20 years. However, accurate lower and upper bounds on the capacity have been determined for some constraints, as discussed in [22], [26], and [27]. For instance, upper bounds for some square checkerboard constraints are given in [27], from which we can conclude that $C(2, 2, 1) \leqslant 0.43431$ and $C(3, 3, 1) \leqslant 0.25681$.

In the rest of this section, we discuss the cases where $\alpha = 1$ or $\beta = 1$. In these cases, the 2-D $(a, b, p) = (\alpha, \beta, p)$-constraint of Definition 2 reduces to a 1-D constraint on each row or column, respectively. We consider first the case where $a = 1$, where the corresponding 1-D constraint is described as follows.

Definition 3: Let b, p be two positive integers. A binary vector $\boldsymbol{u}$ satisfies the (b, p)-*window-weight-limited (WWL) constraint* if for any b consecutive positions there are at most p 1's. We denote the capacity of the constraint by $C_{\mathrm{WWL}}(b, p)$.

According to Theorem 1, $C_{\mathrm{WWL}}(\beta, p)$ is an upper bound on $C(1, \beta, p)$, the capacity of the $(1, \beta, p)$ time–space constraint. Therefore, we are interested in determining the capacity of the general (b, p)-WWL constraint. Before addressing this question, we consider some special cases.

We recall the definition of the (d, k)-runlength-limited (RLL) constraint, which requires that the number of 0's between adjacent 1's is at least d and at most k, and we denote the corresponding capacity by $C_{\mathrm{RLL}}(d, k)$ (see, for example, [16] and [30]). It is easy to see that the $(b, 1)$-WWL constraint is simply the $(b-1, \infty)$-RLL constraint. Therefore, $C_{\mathrm{RLL}}(\beta-1, \infty)$ is an upper bound on $C(1, \beta, 1)$, a result that was already shown by Jiang *et al.* [17].

One can also see that the $(b, b-1)$-WWL constraint is the maximum-transition-run $\mathrm{MTR}(b-1)$ constraint, which limits the maximum length of a run of 1's to no more than $b-1$ [21]. Interchanging the symbols 0 and 1 establishes a one-to-one correspondence between the $\mathrm{MTR}(b-1)$ constraint and the $(0, b-1)$-RLL constraint. Thus, by Theorem 1, $C_{\mathrm{RLL}}(0, \beta-1)$ is an upper bound on $C(1, \beta, \beta-1)$.

The capacity of (d, k)-RLL constraints is well known and can be elegantly described as the logarithm of the largest real root

of a polynomial that depends explicitly on the parameters d and k. (We again refer the reader to [16] and [30].) However, we have not yet found a comparable formulation of the capacity of the general (b, p)-WWL constraint. Therefore, to compute these capacities, we use the general approach that is described in [20].

Definition 4: A *merge* of two vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ of the same length n is a function

$$f_n : \{0,1\}^n \times \{0,1\}^n \mapsto \{0,1\}^{n+1} \cup \{\boldsymbol{F}\}.$$

If the last $n-1$ bits of $\boldsymbol{u}$ are the same as the first $n-1$ bits of $\boldsymbol{v}$, the vector $f_n(\boldsymbol{u}, \boldsymbol{v})$ is the vector $\boldsymbol{u}$ concatenated with the last bit of $\boldsymbol{v}$; otherwise, $f_n(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{F}$.

Definition 5: Let b, p be two positive integers. Let $S_{b,p}$ denote the set of all vectors of length $b-1$ having at most p 1's. That is, $S_{b,p} = \{\boldsymbol{s} \in \{0,1\}^{b-1} : wt(\boldsymbol{s}) \leqslant p\}$. The size of the set $S_{b,p}$ is $M = \sum_{i=0}^{p} \binom{b-1}{i}$. Let $\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_M$ be an ordering of the vectors in $S_{b,p}$. The transition matrix for the (b, p)-WWL constraint, $A_{b,p} = (a_{i,j}) \in \{0,1\}^{M \times M}$, is defined as follows:

$$a_{i,j} = \begin{cases} 1, & \text{if } f_{b-1}(\boldsymbol{s}_i,\ \boldsymbol{s}_j) \neq \boldsymbol{F} \text{ and } wt(f_{b-1}(\boldsymbol{s}_i,\ \boldsymbol{s}_j)) \leqslant p \\ 0, & \text{otherwise.} \end{cases}$$

Example 1: The following illustrates the construction of the transition matrix $A_{3,2}$ associated with the (3, 2)-WWL constraint. Note that

$$S_{3,2} = \{\boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{s}_3, \boldsymbol{s}_4\} = \{(0,0), (0,1), (1,0), (1,1)\}.$$

The merge of $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$ for $i, j = 1, 2, 3, 4$ determines the matrix $A_{3,2}$. For example, $f_2(\boldsymbol{s}_1, \boldsymbol{s}_1) = (0,0,0)$, $a_{1,1} = 1$; $f_2(\boldsymbol{s}_2, \boldsymbol{s}_1) = \boldsymbol{F}$, $a_{2,1} = 0$; $f_2(\boldsymbol{s}_1, \boldsymbol{s}_2) = (0,0,1)$, $a_{1,2} = 1 \neq a_{2,1}$. This shows that the matrix is not necessarily symmetric. Finally, $f_2(\boldsymbol{s}_3, \boldsymbol{s}_3) = (1,1,1)$, and $a_{3,3} = 0$ since (1, 1, 1) does not satisfy the (3,2)-WWL constraint

$$A_{3,2} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

Definition 6: A matrix $A \in \{0,1\}^{M \times M}$ is *irreducible* if for all $1 \leqslant i, j \leqslant M$, there exists some $n \geqslant 0$ such that $(A^n)_{i,j} > 0$. Note that n can be a function of i and j.

Lemma 1: For positive integers b, p, the transition matrix $A_{b,p}$ is irreducible.

Proof: From the construction of $A_{b,p}$, it is clear that $(A^n_{b,p})_{i,j}$ is the number of vectors of length $n+b-1$ starting with $\boldsymbol{s}_i$, ending in $\boldsymbol{s}_j$, and satisfying the (b, p)-WWL constraint, where $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$ are as described in Definition 5. Therefore, $A_{b,p}$ is irreducible if for every pair (i, j), there exists a vector of length $n \geqslant 1$ that starts with $\boldsymbol{s}_i$ and ends in $\boldsymbol{s}_j$. Such a vector is obtained by inserting a sufficient number of 0's between $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$. This proves the irreducibility of $A_{b,p}$. ■

Referring to Theorem 3.9 in [20], we have the following characterization of $C_{\mathrm{WWL}}(b, p)$.

Theorem 2: The capacity of the (b, p)-WWL constraint is given by

$$C_{\mathrm{WWL}}(b, p) = \log_2(\lambda_{\max})$$

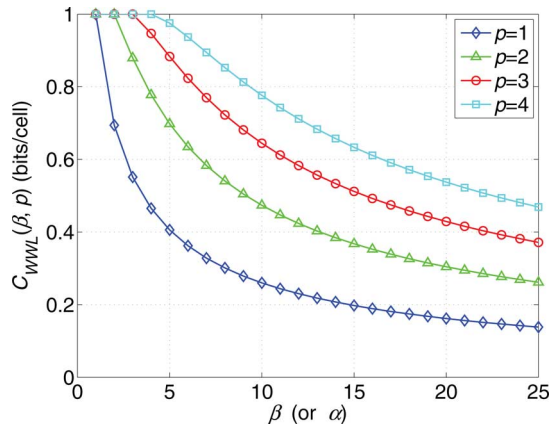

Fig. 1. Upper bound on $C(1, \beta, p)$.

where $\lambda_{\max}$ is the largest real eigenvalue of $A_{b,p}$.

Proof: See Theorem 3.9 in [20]. ■

Fig. 1 shows $C_{\mathrm{WWL}}(\beta, p)$, the upper bound on $C(1, \beta, p)$, for $\beta \leqslant 25$ and $p = 1, 2, 3, 4$. As noted previously, the lowest curve corresponds to the capacity of the $(\beta - 1, \infty)$-RLL constraint.

Remark 2: The construction of the transition matrix $A_{b,p}$ translates into a graph presentation of the (b, p)-WWL constraint in the form of a labeled, directed graph. The states in the graph correspond to the vectors in the set $S_{b,p}$, and the directed edges correspond to the nonzero entries in the matrix $A_{b,p}$. Specifically, if the entry $a_{i,j}$ is nonzero, then there is a directed edge from state $\boldsymbol{s}_i$ to state $\boldsymbol{s}_j$, with label $\boldsymbol{s}_{j,b-1}$, the last bit in $\boldsymbol{s}_j$. Sequences satisfying the (b, p)-WWL constraint are generated by reading off the labels along directed paths in the graph. The graph produced by this construction can be identified with a subgraph of the de Bruijn graph on 2^{b-1} states. Fig. 2 illustrates the graph that generates the (7, 2)-WWL constraint.

Remark 3: According to Theorem 1, the capacity of the (α, p)-WWL constraint, $C_{\mathrm{WWL}}(\alpha, p)$, is an upper bound on the capacity of the $(\alpha, 1, p)$ time–space constraint, $C(\alpha, 1, p)$. Jiang *et al.* [17] proposed an upper bound on the rate of an $(\alpha, 1, 1)$-constrained code with fixed block length n and multiple cell levels. In our numerical experiments, their upper bound for binary cells appears to converge to our upper bound as $n \to \infty$.

IV. Lower Bound on the Capacity

In this section, we give lower bounds on the capacity of the (α, β, p)-constraint based upon specific code constructions. We first present an elementary construction that achieves rate $\frac{p}{\alpha\beta}$. We then show how to improve the bound for the $(1, \beta, p)$-and $(\alpha, 1, p)$-constraints. In this section, we assume that for all positive integers x and y, the value of $x \pmod{y}$ belongs to the group $\{1, \ldots, y\}$ via the correspondence $\{0, 1, \ldots, y-1\} \to \{y, 1, \ldots, y-1\}$.

The idea of Construction 1 is to partition the set of n cells into subblocks of size β. Suppose $p = \beta(q-1) + r$, where $1 \leqslant q \leqslant \alpha$ and $1 \leqslant r \leqslant \beta$. The encoding process has a period of α writes. On the first $q-1$ writes, all cells in each subblock are programmed with no constraint imposed. On the qth write,

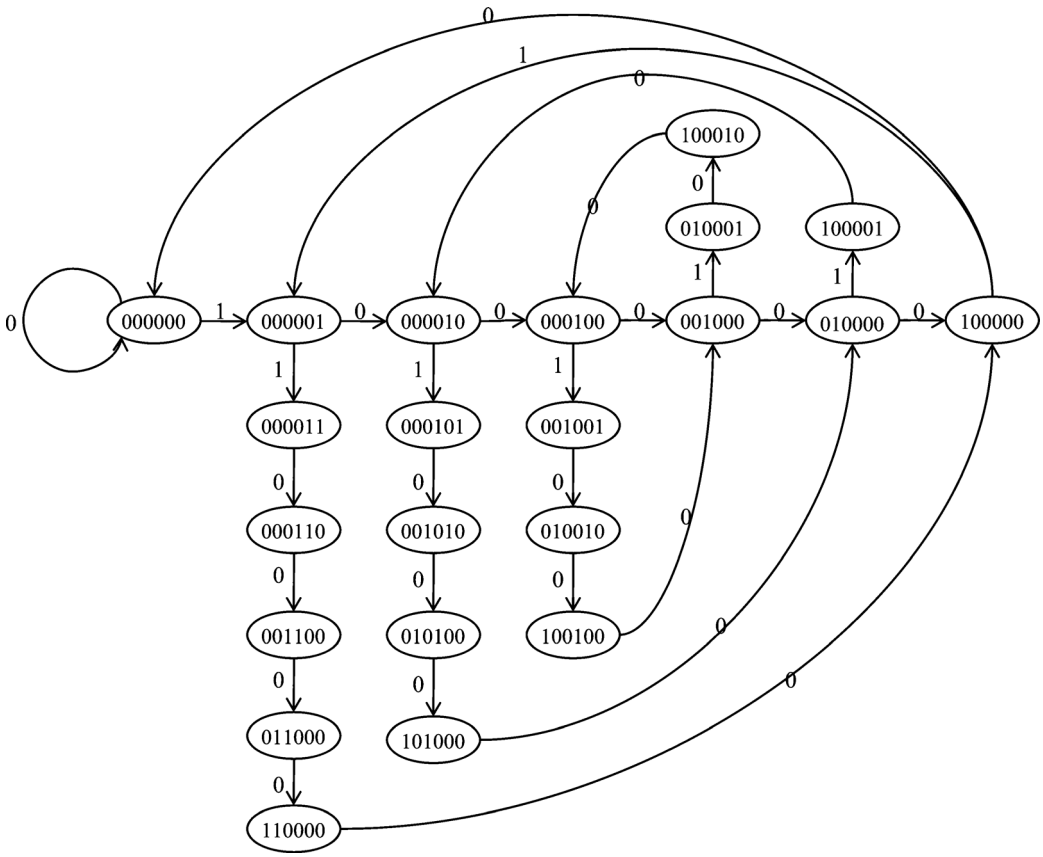

Fig. 2. Labeled graph that generates the (7, 2)-WWL constraint.

the first r cells in each subblock are programmed with no constraint and the rest of the cells are not programmed (staying at level 0). From the $(q+1)$st write to the αth write, no cells are programmed. The details of the construction are as follows.

Construction 1: Let α, β, p be positive integers. We construct an (α, β, p)-constrained code $\mathcal{C}$ of length n as follows. To simplify the construction, we assume that $\beta | n$. Let $q = \left\lceil \frac{p}{\beta} \right\rceil$, $r = p \pmod{\beta}$, where $1 \leqslant r \leqslant \beta$. For all $i \geqslant 1$, on the ith write, the encoder uses the following rules.

1) If $1 \leqslant i \pmod{\alpha} < q$, n bits are written to the n cells.
2) If $i \pmod{\alpha} = q$, rn/β bits are written in all cells c_j such that $1 \leqslant j \pmod{\beta} \leqslant r$.
3) If $i \pmod{\alpha} > q$, no information is written to the cells.

The decoder is implemented in a very similar way.

Example 2: Fig. 3 shows a typical writing sequence of an $(\alpha = 3, \beta = 3, p = 2)$-constrained code of length 15 based on Construction 1. The ith row corresponds to the cell-state vector before the ith write. The cells in the box in the ith row are the only cells that can be programmed on the ith write. It can be seen that the rate of the code is the ratio between the number of boxed cells and the total number of cells, which is $\frac{2}{9}$.

Theorem 3: The code $\mathcal{C}$ constructed in Construction 1 is an (α, β, p)-constrained code and its rate is $R = \frac{p}{\alpha\beta}$.

Proof: We show that for all $i \geqslant 1$ and $1 \leqslant j \leqslant n - \beta + 1$, the rewrite cost of the cells $c_j, c_{j+1}, \ldots, c_{j+\beta-1}$ over the writes $i, i+1, \ldots, i+\alpha-1$, is at most p. For all $0 \leqslant k \leqslant \alpha-1$ such that $1 \leqslant (i+k) \pmod{\alpha} < q$, all of the β cells can be written and since there are $q-1$ such values, the rewrite cost on these writes is at most $(q-1)\beta$. For k, such that $(i+k) \pmod{\alpha} = q$, at most r out of these β cells are programmed, and therefore, the rewrite cost is at most r. For all other values of k, no other cells are programmed. Therefore, the total rewrite cost is at most

$$(q-1)\cdot\beta + r = \left(\left(\left\lceil \frac{p}{\beta} \right\rceil - 1\right)\beta + p\right) \pmod{\beta} = p.$$

The total number of bits written on these α writes is pn/β, and hence, the rate of the code is

$$R = \frac{pn/\beta}{\alpha n} = \frac{p}{\alpha\beta}.$$

■

0: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1: 1 1 0 0 1 0 1 0 0 0 0 0 0 1 0
2: 1 1 0 0 1 0 1 0 0 0 0 0 0 1 0
3: 1 1 0 0 1 0 1 0 0 0 0 0 0 1 0
4: 1 0 0 1 0 0 1 1 0 0 1 0 1 1 0
5: 1 0 0 1 0 0 1 1 0 0 1 0 1 1 0
6: 1 0 0 1 0 0 1 1 0 0 1 0 1 1 0
7: 0 0 0 0 1 0 1 0 0 1 1 0 0 1 0
8: 0 0 0 0 1 0 1 0 0 1 1 0 0 1 0

Fig. 3. Sequence of writes of a (3, 3, 2)-constrained code.

A. Space Constraint Improvement

In this section, we improve upon the lower bound on $C(1, \beta, p)$ obtained from the elementary construction. Let $\mathcal{S}_n(b, p)$ be the set of all (b, p)-WWL vectors of length n. We refer to a subset of $\mathcal{S}_n(b, p)$ as a (b, p)-*WWL code* $\mathcal{C}_{\mathrm{WWL}}$ of

length n. If the size of the code $\mathcal{C}_{\mathrm{WWL}}$ is M, then it is specified by an encoding map $\mathcal{E}_{\mathrm{WWL}} : \{1,\ldots,M\} \mapsto \mathcal{C}_{\mathrm{WWL}}$ and a decoding map $\mathcal{D}_{\mathrm{WWL}} : \mathcal{C}_{\mathrm{WWL}} \mapsto \{1,\ldots,M\}$, such that for all $m \in \{1,\ldots,M\}$, $\mathcal{D}_{\mathrm{WWL}}(\mathcal{E}_{\mathrm{WWL}}(m)) = m$.

The problem of finding (b,p)-WWL codes that approach or achieve the capacity $C_{\mathrm{WWL}}(b,p)$ is of independent interest and we address it next. Cover [7] provided an enumerative scheme that can be used to calculate the lexicographic order of any sequence in the constrained system. For the special case of $p=1$, corresponding to RLL block codes, Datta and McLaughlin [8], [9] proposed enumerative methods for binary (d,k)-RLL codes based on permutation codes. For (b,p)-WWL codes, we find enumerative encoding and decoding strategies with linear complexity enumerating all (b,p)-WWL vectors. We present the coding schemes and the complexity analysis in Appendix A. In the sequel, we will simply assume that there exist such codes with rate arbitrarily close to the capacity as the block length goes to infinity for all positive integers b and p. The next construction uses (β,p)-WWL codes to construct $(1,\beta,p)$-constrained codes.

Construction 2: Let β, p be positive integers such that $p \leqslant \beta$. Let $\mathcal{C}_{\mathrm{WWL}}$ be a (β,p)-WWL code of length n' and size M. Let $\mathcal{E}_{\mathrm{WWL}}$ and $\mathcal{D}_{\mathrm{WWL}}$ be its encoding and decoding maps. A $(1,\beta,p)$-constrained code $\mathcal{C}_{1,\beta,p}$ of length $n = 2n' + \beta - 1$ and its encoding map $\mathcal{E}$ and decoding map $\mathcal{D}$ are constructed as follows.

1) The encoding map $\mathcal{E} : \{1,\ldots,M\} \times \{0,1\}^n \to \{0,1\}^n$ is defined for all $(m,\boldsymbol{u}) \in \{1,\ldots,M\} \times \{0,1\}^n$ to be $\mathcal{E}((m,\boldsymbol{u})) = \boldsymbol{v}$, where
   a) $\boldsymbol{v}_1^{n'} = \boldsymbol{u}_1^{n'} + \mathcal{E}_{\mathrm{WWL}}(m)$;
   b) $\boldsymbol{v}_{n'+1}^{n'+\beta-1} = \boldsymbol{0}$;
   c) $\boldsymbol{v}_{n'+\beta}^{n} = \boldsymbol{u}_1^{n'}$.
2) The decoding map $\mathcal{D} : \{0,1\}^n \to \{1,\ldots,M\}$ is defined for all $\boldsymbol{u} \in \{0,1\}^n$ to be

$$\mathcal{D}(\boldsymbol{u}) = \mathcal{D}_{\mathrm{WWL}}(\boldsymbol{v}_1^{n'} + \boldsymbol{v}_{n'+\beta}^{n}).$$

Example 3: Here is an example of an $(\alpha=1,\beta=3,p=2)$ code with $n'=4$ for the first 4 writes. The message set has size $M_{n'}=13$ (see the definition of $M_{n'}$ in Definition 7 in Appendix A). The length of the memory is $2n'+\beta-1=10$. Suppose on the second write, the message is $m=7$. Since lexicographically the seventh element in $\mathcal{S}_4(3,2)$ is (0110), the encoder will copy the previous left block (1011) to the right block and flip the second and the third bits in the left block $(1011) \to (1101)$

		0	0	0	0	0	0	0	0	0	0
1st write,	$m=11$:	**1**	0	**1**	**1**	0	0	0	0	0	0
2nd write,	$m=7$:	1	**1**	**0**	1	0	0	**1**	0	**1**	**1**
3rd write,	$m=13$:	**0**	**0**	0	**0**	0	0	1	**1**	**0**	1
4th write,	$m=4$:	0	0	**1**	**1**	0	0	**0**	**0**	0	**0**.

Theorem 4: The code $\mathcal{C}_{1,\beta,p}$ is a $(1,\beta,p)$-constrained code. If the rate of the code $\mathcal{C}_{\mathrm{WWL}}$ is R_{WWL}, then the rate of the code $\mathcal{C}_{1,\beta,p}$ is $\frac{n'}{2n'+\beta-1} \cdot R_{\mathrm{WWL}}$. Both the encoder and decoder of $\mathcal{C}_{1,\beta,p}$ have complexity $O(n)$.

Proof: Let $\boldsymbol{u}$ be the cell-state vector in Construction 2.

1) For $\boldsymbol{u}_1^{n'}$, encoder step a) guarantees that the positions of rewritten cells satisfy the (β,p)-WWL constraint. So there are at most p reprogrammed cells in any β consecutive cells in $\boldsymbol{u}_1^{n'}$.
2) For $\boldsymbol{u}_{n'+\beta}^{n}$, three consecutive writes should be examined. Let $\boldsymbol{w},\boldsymbol{v},\boldsymbol{u}$ be the cell-state vectors before the ith, $(i+1)$st, $(i+2)$nd writes, $i \geqslant 1$. Encoder step a) means that $\boldsymbol{v}_1^{n'} = \boldsymbol{w}_1^{n'} + \mathcal{E}_{\mathrm{WWL}}(m_i)$, where $m_i \in \{1,\ldots,M\}$ is the message to encode on the ith write. Since encoder step c) guarantees that $\boldsymbol{v}_{n'+\beta}^{n} = \boldsymbol{w}_1^{n'}$ and $\boldsymbol{u}_{n'+\beta}^{n} = \boldsymbol{v}_1^{n'}$, we have $\boldsymbol{u}_{n'+\beta}^{n} = \boldsymbol{v}_{n'+\beta}^{n} + \mathcal{E}_{\mathrm{WWL}}(m_i)$. This proves that $\boldsymbol{u}_{n'+\beta}^{n}$ satisfies the $(1,\beta,p)$ constraint.
3) For $\boldsymbol{u}_{n'+1}^{n'+\beta-1}$, the cell levels are always set to be 0, which ensures that no violation of the constraint happens between $\boldsymbol{u}_1^{n'}$ and $\boldsymbol{u}_{n'+\beta}^{n}$.

On each write, one of M messages is encoded as a vector of length n. Hence, the rate is $\frac{\log_2 M}{n} = \left(\frac{\log_2 M}{n'}\frac{n'}{2n'+\beta-1}\right) = \frac{n'}{2n'+\beta-1} \cdot R_{\mathrm{WWL}}$.

The encoder $\mathcal{E}$ and decoder $\mathcal{D}$ come directly from $\mathcal{E}_{\mathrm{WWL}}$ and $\mathcal{D}_{\mathrm{WWL}}$, which have complexity $O(n)$ both in time and in space. Therefore, $\mathcal{E}$ and $\mathcal{D}$ both have linear complexity in time and in space. ■

Corollary 1: Let β, p be two positive integers such that $p \leqslant \beta$; then

$$C(1,\beta,p) \geqslant \max\left\{\frac{C_{\mathrm{WWL}}(\beta,p)}{2}, \frac{p}{\beta}\right\}.$$

Corollary 1 provides a lower bound that is achieved by practical coding schemes. In fact, following similar proofs in [3], [5], and [6], we can prove the following theorem using probabilistic combinatorial tools [2].

Theorem 5: Let β, p be positive integers such that $\beta \geqslant p$. Then

$$C(1,\beta,p) = C_{\mathrm{WWL}}(\beta,p).$$

Proof: See Appendix B. ■

B. Time Constraint Improvement

Jiang *et al.* constructed in [17] an $(\alpha,1,1)$-constrained code. Let us explain their construction as it serves as the basis for our construction. Their construction uses write-once memory (WOM) codes [24]. A WOM is a storage device consisting of cells that can be used to store any of q values. In the binary case, each cell can be irreversibly changed from state 0 to state 1. We denote by $[n,t;2^{nR_1},\ldots,2^{nR_t}]$ a t-write WOM code $\mathcal{C}_W$ such that the number of messages that can be written to the memory on its ith write is 2^{nR_i}, and the sum-rate of the WOM code is defined to be $R_{\mathrm{sum}} = \sum_{i=1}^{t} R_i$. The sum-capacity C_{sum} is defined as the supremum of achievable sum-rates. The code is specified by t pairs of encoding and decoding maps, $(\mathcal{E}_i,\mathcal{D}_i)$, where $i \in \{1,2,\ldots,t\}$. Assuming that the cell-state vector before the ith write is $\boldsymbol{c}_i$, the encoder is a map

$$\mathcal{E}_i : [1 : 2^{nR_i}] \times \{0,1\}^n \to \{0,1\}^n$$

such that for all $(m,\boldsymbol{c}_{i-1}) \in [1 : 2^{nR_i}] \times \{0,1\}^n$

$$\boldsymbol{c}_{i-1} \preceq \boldsymbol{c}_i = \mathcal{E}_i(m,\boldsymbol{c}_{i-1})$$

TABLE I
3-CELL 2-WRITE WOM CODE

message	1st write	2nd write
0	000	111
1	001	110
2	010	101
3	100	011

where the relation "$\preceq$" is defined in Definition 7. The decoder

$$\mathcal{D}_i : \{0,1\}^n \to [1 : 2^{nR_i}]$$

satisfies

$$\mathcal{D}_i(\mathcal{E}_i(m, \boldsymbol{c}_{i-1})) = m$$

for all $m \in [1 : 2^{nR_i}]$.

It has been shown in [14] that the sum-capacity of a t-write WOM is $C_{\text{sum}} = \log_2(t+1)$.

Example 4: Table I shows the encoding and decoding maps of a 2-write WOM code using three cells, where on each write, 2 bits are written [24]. Suppose all cells are initialized as 0. If the written messages are 1 and 3 on the first and the second write, respectively, then the cell-state vector is changed as $(000) \to (001) \to (011)$; if on both writes, message 2 is written, then the cell-state vector is changed as $(000) \to (010) \to (010)$.

The constructed $(\alpha, 1, 1)$-constrained code has a period of $2(t+\alpha)$ writes. On the first t writes of each period, the encoder simply writes the information using the encoding maps of the t-write WOM code. Then, on the $(t+1)$st write, no information is written but all the cells are increased to level one. In the following $\alpha - 1$ writes, no information is written and the cells do not change their levels; that completes half of the period. On the next t writes, the same WOM code is again used; however, since now all the cells are in level one, the complement of the cell-state vector is written to the memory on each write. On the next write, no information is written and the cells are reduced to level zero. In the last $\alpha - 1$ writes, no information is written and the cells do not change their values. We present this construction now in detail.

Construction 3: Let α be a positive integer and let $\mathcal{C}_W$ be an $[n, t; 2^{nR_1}, \ldots, 2^{nR_t}]$ t-write WOM code. Let $\mathcal{E}_i(m, \boldsymbol{v}_{i-1})$ be the ith encoder of $\mathcal{C}_W$, for $m \in [1 : 2^{nR_i}]$, $i \in [1 : t]$. An $(\alpha, 1, 1)$-constrained code $\mathcal{C}_{\alpha,1,1}$ is constructed as follows. For all $i \geqslant 1$, let $i' = i \pmod{2(t+\alpha)}$, where $1 \leqslant i' \leqslant 2(t+\alpha)$. The cell-state vector after the ith write is denoted by $\boldsymbol{c}_i$. On the ith write, the encoder uses the following rules.

1) If $i' \in [1 : t]$, write $M_{i'} \in [1 : 2^{nR_{i'}}]$ such that
$$\boldsymbol{c}_i = \mathcal{E}_{i'}(M_{i'}, \boldsymbol{c}_{i-1}).$$
2) If $i' = t+1$, no information is written and the cell-state vector is changed to the all-one vector $\mathbf{1}$, i.e., $\boldsymbol{c}_i = \mathbf{1}$.
3) If $i' \in [t+2 : t+\alpha]$, no information is written and the cell-state vector is not changed.
4) If $i' \in [t+\alpha+1 : 2t+\alpha]$, write $M_{i'-t-\alpha} \in [1 : 2^{nR_{i'-t-\alpha}}]$ such that
$$\boldsymbol{c}_i = \overline{\mathcal{E}_{i'-t-\alpha}(M_{i'-t-\alpha}, \overline{\boldsymbol{c}}_{i-1})}.$$
5) If $i' = 2t+\alpha+1$, no information is written and the cell-state vector is changed to the all-zero vector $\mathbf{0}$, i.e., $\boldsymbol{c}_i = \mathbf{0}$.
6) If $i' \in [2t+\alpha+1 : 2(t+\alpha)]$, no information is written and the cell-state vector is not changed.

Remark 4: This construction is presented differently in [17]. This results from the constraint of having the same rate on each write which we can bypass in this work. Consequently, in our case, we can have varying rates, and thus, the code $\mathcal{C}_{\alpha,1,p}$ can achieve a higher rate.

Theorem 6: The code $\mathcal{C}_{\alpha,1,1}$ is an $(\alpha, 1, 1)$-constrained code. If the t-write WOM code $\mathcal{C}_W$ is sum-rate optimal, then the rate of $\mathcal{C}_{\alpha,1,1}$ is $\frac{\log_2(t+1)}{t+\alpha}$.

Proof: In every period of $2(t+\alpha)$ writes, every cell is programmed at most twice: once in the first $t+1$ writes and once in the first $t+1$ writes of the second part of the write-period. After every sequence of $t+1$ writes, the cell is not programmed for $\alpha - 1$ writes. Therefore, the rewrite cost of every cell among α consecutive rewrites is at most 1.

If the rate of the WOM code $\mathcal{C}_W$ is R_W, then $2nR_W$ bits are written in every period of $2(t+\alpha)$ writes. Hence, the rate of $\mathcal{C}_{\alpha,1,1}$ is $\frac{2nR_W}{2(t+\alpha)n} = \frac{R_W}{t+\alpha}$. If $\mathcal{C}_W$ is sum-rate optimal, the rate of $\mathcal{C}_{\alpha,1,1}$ is therefore $\frac{\log_2(t+1)}{t+\alpha}$. ∎

The next table shows the highest rates of $(\alpha, 1, 1)$-constrained codes based on Construction 3 for $\alpha = 4, \ldots, 8$

α	4	5	6	7	8
$1/\alpha$	0.25	0.2	0.167	0.143	0.125
rate of $\mathcal{C}_{\alpha,1,1}$	0.290	0.256	0.235	0.216	0.201

.

Next, we would like to extend Construction 3 in order to construct $(\alpha, 1, p)$-constrained codes for all $p \geqslant 2$. For simplicity of the construction, we will assume that p is an even integer; the required modification for odd values of p will be immediately clear. We choose $t \geqslant 1$ such that $\alpha \geqslant (p-1)t$ and the period of the code is $\alpha + t$. On the first t writes of each period, the encoder uses the encoding map of the t-write WOM code. In the following t writes, it uses the bitwise complement of a WOM code as in Construction 3. This procedure is repeated for $\frac{p}{2}$ times; this completes the first tp writes in the period. On the $(tp+1)$st write, no new information is written and the cell-state vector is changed to the all-zero vector. During the $(tp+2)$nd to $(\alpha+t)$th writes, no information is written and the cell-state vector is not changed. That completes one period of $\alpha+t$ writes.

Remark 5: If p is odd, then on the $(tp+1)$st write, no new information is written and the cell-state vector is changed to the all-one vector. It is not changed until the $(\alpha+t)$th write to complete a period. Now the cell-state vector is an all-one vector. For the next period of $(\alpha+t)$ writes, the encoder uses the bitwise complement of the first period and the cell-state vector returns to all-zero state afterward.

Construction 4: Let α, p, t be positive integers such that $\alpha \geqslant (p-1)t$ and p is even. Let $\mathcal{C}_W$ be an $[n, t; 2^{nR_1}, \ldots, 2^{nR_t}]$ t-write WOM code. For $i \in [1 : t]$, let $\mathcal{E}_i(m, \boldsymbol{v}_{i-1})$ be its encoding map on the ith write, where $m \in [1 : 2^{nR_i}]$. An $(\alpha, 1, p)$-constrained code $\mathcal{C}_{\alpha,1,p}$ is constructed as follows. For all $i \geqslant 1$, let $i' = i \pmod{\alpha + t}$,

$i'' = i' \pmod{2t}$ where $1 \leqslant i' \leqslant (\alpha + t)$, $1 \leqslant i'' \leqslant 2t$. The cell-state vector after the ith write is denoted by $\boldsymbol{c}_i$. On the ith write, the encoder uses the following rules.

1) If $i' \in [1 : pt]$ and $i'' \in [1 : t]$, write $M_{i''} \in [1 : 2^{nR_{i''}}]$ such that
$$\boldsymbol{c}_i = \mathcal{E}_{i''}(M_{i''}, \boldsymbol{c}_{i-1}).$$
2) If $i' \in [1 : pt]$ and $i'' \in [t+1 : 2t]$, write $M_{i''-t} \in [1 : 2^{nR_{i''-t}}]$ such that
$$\boldsymbol{c}_i = \overline{\mathcal{E}_{i''-t}(M_{i''-t}, \overline{\boldsymbol{c}}_{i-1})}.$$
3) If $i' = pt + 1$, no information is written and the cell-state vector is changed to $\mathbf{0}$, i.e., $\boldsymbol{c}_i = \mathbf{0}$.
4) If $i' \in [pt + 2 : \alpha + t]$, no information is written and the cell-state vector is not changed.

Example 5: Suppose $\alpha = 3$, $p = 2$, and $t = 2$ in Construction 4 and $\mathcal{C}_W$ is the WOM code in Example 4. The period of Construction 4 is $\alpha + t = 5$. Suppose all cells are initialized as 0 and the messages to write are $(1, 3, 2, 1)$ on the first 4 writes, and no information is written on the fifth write. Then the cell-state vector is changed as $(000) \to (001) \to (011) \to (101) \to (001) \to (000)$.

Theorem 7: The code $\mathcal{C}_{\alpha,1,p}$ is an $(\alpha, 1, p)$-constrained code. If the t-write WOM code $\mathcal{C}_W$ is sum-rate optimal, then the rate of $\mathcal{C}_{\alpha,1,p}$ is $\frac{p \log_2(t+1)}{\alpha+t}$. □

Proof: This is similar to the proof of Theorem 6, so we present here only a sketch of the proof. In every period of $(\alpha+t)$ writes, each cell is rewritten at most p times. In particular, the first rewrite happens before the $(t+1)$st write. After that, the cell is rewritten at most $p-1$ times until the $(tp+1)$st write and then not programmed for $\alpha+t-(tp+1)$ writes. Therefore, each cell is rewritten at most p times on $\alpha+t-(tp+1)+(tp+1)-t = \alpha$ writes. This proves the validity of the code.

If the rate of the WOM code $\mathcal{C}_W$ is R_W, then pnR_W bits are written during each period of $\alpha + t$ writes since the WOM code is used p times. Hence, the rate of $\mathcal{C}_{\alpha,1,p}$ is $\frac{2pnR_W}{2(\alpha+t)n} = \frac{pR_W}{\alpha+t}$. If $\mathcal{C}_W$ is sum-rate optimal, the rate of $\mathcal{C}_{\alpha,1,p}$ is $\frac{p \log_2(t+1)}{\alpha+t}$. ■

Remark 6: In Construction 4, we required that $\alpha \geqslant (p-1)t$ and, in particular, $t \leqslant \left\lfloor \frac{\alpha}{p-1} \right\rfloor$. If $t > \left\lfloor \frac{\alpha}{p-1} \right\rfloor$, we can simply use Construction 4 while taking $\alpha = (p-1)t$, i.e., the period of writes is now pt and we construct a $((p-1)t, 1, p)$-constrained code, which is also an $(\alpha, 1, p)$-constrained code. The rate of the code is R_W/t, where R_W is the rate of the WOM code $\mathcal{C}_W$.

The next corollary provides lower bounds on $C(\alpha, 1, p)$.

Corollary 2: Let α, p be positive integer such that $p \leqslant \alpha$. Then
$$C(\alpha, 1, p) \geqslant \max_{t, t^* \in \mathbb{Z}_+,} \left\{ \frac{p \log_2(t+1)}{\alpha + t}, \frac{\log_2(t^*+1)}{t^*}, \frac{p}{\alpha} \right\}$$
where
$$1 \leqslant t \leqslant \left\lfloor \frac{\alpha}{p-1} \right\rfloor, t^* = \left\lceil \frac{\alpha}{p-1} \right\rceil.$$

Fig. 4 shows the rates of $(\alpha, \beta = 1, p)$-constrained codes obtained by selecting the best t for each pair of (α, p). Note that the curve for $p = 1$ is obtained by implementing the ideas in [17] and there is a slight improvement over the rates listed in [17] for $\alpha = 4, \ldots, 8$, the reason for which is discussed in Remark 4. In comparison to the codes in Construction 1, whose rates are shown by the dashed lines, our construction approximately doubles the rates. Our lower bounds achieve approximately 78% of the corresponding upper bounds on $C(\alpha, 1, p)$.

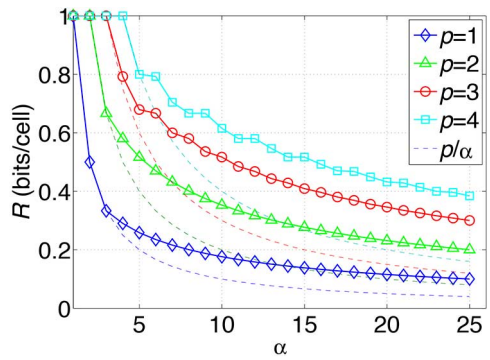

Fig. 4. Lower bound on $C(\alpha, 1, p)$.

C. Time–Space Constraint Improvement

In this section, we are interested in combining the improvements in time and in space to provide lower bounds on the capacity of (α, β, p)-constraints.

Theorem 8: For all α, β, p positive integers
$$C(\alpha, \beta, p) \geqslant \max \left\{ \frac{C(\alpha, 1, p)}{\beta}, \frac{C(1, \beta, p)}{\alpha} \right\}.$$

Proof: An (α, β, p)-constrained code can be constructed in two ways.

1) Let $\mathcal{C}$ be a $(1, \beta, p)$-constrained code of rate R and length n. We construct a new code $\mathcal{C}'$ with the same number of cells. New information is written to the memory on all ith writes, where $i \equiv 1 \pmod{\alpha}$, simply by using the $\lceil \frac{i}{\alpha} \rceil$th write of the code $\mathcal{C}$. Then, the code $\mathcal{C}'$ is an (α, β, p)-constrained code and its rate is R/α. Therefore, we conclude that $C(\alpha, \beta, p) \geqslant \frac{C(1,\beta,p)}{\alpha}$.
2) Let $\mathcal{C}$ be an $(\alpha, 1, p)$-constrained code of rate R and length n. We construct a new code $\mathcal{C}'$ for $n\beta$ cells: $(c_1, c_2, \ldots, c_{n\beta})$. The code $\mathcal{C}'$ uses the same encoding and decoding maps of the code $\mathcal{C}$, while using only the n cells c_i such that $i \equiv 1 \pmod{\beta}$. Then, the code $\mathcal{C}'$ is an (α, β, p)-constrained code and its rate is R/β. Therefore, we conclude that $C(\alpha, \beta, p) \geqslant \frac{C(\alpha,1,p)}{\beta}$.

The capacity must be greater than or equal to the maximum of the two lower bounds. ■

APPENDIX A

In this section, we show an enumerative encoding and decoding strategy with linear complexity for the set of (β, p)-WWL vectors.

Definition 7: Let $\mathbf{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ be a set of distinct binary vectors, $\boldsymbol{x}_i \in \{0, 1\}^n$, $i = 1, \ldots, N$. Let $\psi(\boldsymbol{x})$ denote the decimal representation of a vector $\boldsymbol{x} \in \{0, 1\}^n$. For $\boldsymbol{x}, \boldsymbol{y} \in \{0, 1\}^n$, we say $\boldsymbol{x} \preceq \boldsymbol{y}$ (or $\boldsymbol{x} \prec \boldsymbol{y}$) if and only if $\psi(\boldsymbol{x}) \leqslant \psi(\boldsymbol{y})$

(or $\psi(\boldsymbol{x}) < \psi(\boldsymbol{y})$). The order of the element $\boldsymbol{x}_i$ in $\mathbf{X}$ is defined as

$$ord(\boldsymbol{x}_i) = \big|\{j : \boldsymbol{x}_j \preceq \boldsymbol{x}_i, 1 \leqslant j \leqslant N\}\big|.$$

Let $\{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_{M_n}\}$ be an ordering of the elements in $\mathcal{S}_n(\beta, p)$, where $M_n = |\mathcal{S}_n(\beta, p)|$. The encoder and decoder of a (β, p)-WWL code give a one-to-one mapping between $\mathcal{S}_n(\beta, p)$ and $\{1, \ldots, M_n\}$, namely $\mathcal{E}_{\mathrm{WWL}}(m) = \boldsymbol{c}_m$ where $ord(\boldsymbol{c}_m) = m$ and $\mathcal{D}_{\mathrm{WWL}}(\boldsymbol{c}_m) = ord(\boldsymbol{c}_m) = m$, for all $m = \{1, \ldots, M_n\}$. Now the problem is to calculate $ord(\boldsymbol{c}_m)$ given $\boldsymbol{c}_m$. Let $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_{M_{\beta-1}}$ be the ordering of the vectors in $S_{\beta,p}$ introduced in Definition 5, where $M_{\beta-1} = |S_{\beta,p}| = |\mathcal{S}_{\beta-1}(\beta, p)| = \sum_{i=0}^{p} \binom{\beta-1}{i}$. Let

$$\boldsymbol{x}_{\beta,p,n} = (x_1(n), x_2(n), \ldots, x_{M_{\beta-1}}(n))^T$$

where $x_i(n)$ is the number of (β, p)-WWL vectors of length n that have the vector $\boldsymbol{s}_i$ as a prefix, where $\boldsymbol{x}^T$ denotes the transpose of $\boldsymbol{x}$.

Lemma 2: The vectors $\boldsymbol{x}_{\beta,p,n+1}$, $n \geqslant \beta$, satisfy the first-order recursion

$$\boldsymbol{x}_{\beta,p,n+1} = A_{\beta,p} \cdot \boldsymbol{x}_{\beta,p,n}.$$

Proof: See [27]. ■

The encoder and decoder have access to a matrix $\boldsymbol{X}_{\beta,p,n} \in \mathbf{Z}_{+}^{(n+\beta)\times M_{\beta-1}}$, where the ith row of $\boldsymbol{X}_{\beta,p,n}$ is $\boldsymbol{x}_{\beta,p,i}^T$, $i = 1, \ldots, n+\beta$. For simplicity, $\boldsymbol{X}_{\beta,p,n}$ is written as $\boldsymbol{X}$ if no confusion can occur. We denote by $\boldsymbol{X}(i, j)$ the entry in the ith row and jth column of $\boldsymbol{X}$ and we define $\boldsymbol{X}(i, :)$, $\boldsymbol{X}(:, j)$ to be the ith row vector, jth column vector of $\boldsymbol{X}$, respectively, i.e., $\boldsymbol{X}(i, :) = (\boldsymbol{X}(i, 1), \ldots, \boldsymbol{X}(i, M_{\beta-1}))$ and $\boldsymbol{X}(:, j) = (\boldsymbol{X}(1, j), \ldots, \boldsymbol{X}(n+\beta, j))^T$. From Lemma 2, $\boldsymbol{X}_{\beta,p,n}$ can be calculated efficiently with time complexity $O(n)$.

1) Decoder: Based on $\boldsymbol{X}_{\beta,p,n}$, we present an enumerative method to calculate the order of each element in $\mathcal{S}_n(\beta, p)$. Note that the order of a vector is the decoded message corresponding to that vector. In this algorithm, the decoder scans the vector from left to right. Whenever the decoder finds a 1 in the vector, the order of the vector will increase. The details of the algorithm are presented below. Here, $\boldsymbol{c} = (c_1, \ldots, c_n) \in \mathcal{S}_n(\beta, p)$ is the binary vector to be decoded; the algorithm calculates $ord(\boldsymbol{c}) \in \{1, \ldots, M_n\}$.

Algorithm 1 Decoding: Calculate $ord(\boldsymbol{c})$, $\boldsymbol{c} \in \mathcal{S}_n(\beta, p)$

```
1:  let cnt = 0, j = 1, i = 0;
2:  while (i ⩽ n){
3:    while (j ⩽ n and c(j) ≠ 1)
4:      j = j + 1;
5:    if (j = n + 1)
6:      ord(c) = cnt + 1;
7:      algorithm ends;
8:    }
9:    /*A 1 is detected in c.*/
10:   let d = (0, ..., 0) with length β − 1;
11:/*d is a vector storing β − 2 bits to the left of the detected
    1, with a 0 appended.*/
12:   if (j ⩾ β − 1)
13:     let d_1^{β−2} = c_{j−β+2}^{j−1};
14:   else/*j < β − 1*/
15:     let d_{β−j}^{β−2} = c_1^{j−1};
16:   find k ∈ [1 : M_{β−1}] such that s_k = d;
17:   cnt = cnt + X(n − j + β − 1, k);
18:   i = j; j = i + 1;
19: }
20: ord(c) = cnt + 1;
21: algorithm ends.
```

Example 6: Suppose we would like to decode a $(6, 3)$-WWL vector $\boldsymbol{c} = (1011001001)$ of length 10.

1) A 1 is detected ($\mathbf{1}011001001$), where $i = 0$ and $j = 1$. The decoder aims to find the number of vectors $\hat{\boldsymbol{c}}$ such that $(0000000000) \preceq \hat{\boldsymbol{c}} \prec (1000000000)$. Now $\boldsymbol{d} = (00000) = \boldsymbol{s}_1$, so $k = 1$, and $n - j + \beta - 1 = 14$. Therefore, $cnt = 0 + \boldsymbol{X}_{6,3,16}(14, 1) = 236$.
2) A 1 is detected ($10\mathbf{1}1001001$), where $i = 1$ and $j = 3$. The decoder aims to find the number of vectors $\hat{\boldsymbol{c}}$ such that $(1000000000) \preceq \hat{\boldsymbol{c}} \prec (1010000000)$. Here, $\boldsymbol{d} = (00100) = s_5$, so $k = 5$, and $n - j + \beta - 1 = 12$. Therefore, $cnt = 236 + \boldsymbol{X}_{6,3,16}(12, 5) = 308$.
3) A 1 is detected ($101\mathbf{1}001001$), where $i = 3$ and $j = 4$. The decoder aims to find the number of vectors $\hat{\boldsymbol{c}}$ such that $(1001000000) \preceq \hat{\boldsymbol{c}} \prec (1011000000)$. Here, $\boldsymbol{d} = (01010) = s_{11}$, so $k = 11$, and $n - j + \beta - 1 = 11$. Therefore, $cnt = 308 + \boldsymbol{X}_{6,3,16}(11, 11) = 343$.
4) A 1 is detected ($1011001\mathbf{1}001$), where $i = 4$ and $j = 7$. The decoder aims to find the number of vectors $\hat{\boldsymbol{c}}$ such that $(1011000000) \preceq \hat{\boldsymbol{c}} \prec (1011001000)$. Here, $\boldsymbol{d} = (11000) = s_{23}$, so $k = 23$, and $n - j + \beta - 1 = 8$. Therefore, $cnt = 343 + \boldsymbol{X}_{6,3,16}(8, 23) = 351$.
5) Finally, a 1 is detected ($101100100\mathbf{1}$), where $i = 7$ and $j = 10$. The decoder aims to find the number of vectors $\hat{\boldsymbol{c}}$ such that $(1011001000) \preceq \hat{\boldsymbol{c}} \prec (1011001001)$. Here, $\boldsymbol{d} = (01000) = s_9$, so $k = 9$, and $n - j + \beta - 1 = 5$. Therefore, $cnt = 351 + \boldsymbol{X}_{6,3,16}(5, 9) = 352$.

We calculate that $ord(\boldsymbol{c}) = cnt + 1 = 353$ and $\boldsymbol{c}$ is decoded as 353.

Theorem 9: Algorithm 1 calculates the order of a (β, p)-WWL vector of length n in $\mathcal{S}_n(\beta, p)$. Its time complexity and space complexity are both $O(n)$. □

Proof: We first show the correctness of the algorithm and then analyze its time and space complexity.

Correctness: Let $\boldsymbol{c}$ be the vector to decode; that is, we seek to find $ord(\boldsymbol{c})$. For $\boldsymbol{c}_1 \preceq \boldsymbol{c}_2$, we denote by $N(\boldsymbol{c}_1, \boldsymbol{c}_2)$ the number of vectors $\hat{\boldsymbol{c}}$ such that $\boldsymbol{c}_1 \preceq \hat{\boldsymbol{c}} \prec \boldsymbol{c}_2$. Let $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_L$ be a sequence of vectors such that $\boldsymbol{0} = \boldsymbol{c}_0 \preceq \boldsymbol{c}_1 \preceq \boldsymbol{c}_2 \preceq \cdots \preceq \boldsymbol{c}_L = \boldsymbol{c}$; then, it is easy to see

$$ord(\boldsymbol{c}) = \sum_{i=1}^{L} N(\boldsymbol{c}_{i-1}, \boldsymbol{c}_i) + 1.$$

Let L be the number of 1's in $\boldsymbol{c}$; let all the indices of 1's be $j_1, j_2, \ldots, j_L$ in ascending order, that is, $1 \leqslant j_1 < \cdots < j_L \leqslant n$ and $c_{j_1} = c_{j_2} = \cdots = c_{j_L} = 1$. For $i \in \{1, \ldots, L\}$, $\boldsymbol{c}_i$ is chosen such that $\boldsymbol{c}_i = \boldsymbol{c}_{i-1} + \boldsymbol{\delta}_{j_i}$, where $\boldsymbol{c}_0 = \boldsymbol{0}$, and $\boldsymbol{\delta}_j$, $j \in \{1, \ldots, n\}$, denotes the vector where all entries are 0 except for the jth entry, which is a 1. Here, addition is componentwise modulo-2 summation.

Lines 3 and 4 together with Line 18 in Algorithm 1 scan $\boldsymbol{c}$ and find $\boldsymbol{c}_i$ according to $\boldsymbol{c}_{i-1}$. Therefore, we are left to prove that Algorithm 1 calculates $N(\boldsymbol{c}_{i-1}, \boldsymbol{c}_i)$ for $i \in \{1, \ldots, L\}$.

By definition, the first $j_i - 1$ digits of $\boldsymbol{c}_i$ and $\boldsymbol{c}_{i-1}$ are the same, and $c_{i,j_i} = 1$ while $c_{i-1,j_i} = 0$. Then, a vector $\hat{\boldsymbol{c}} \in \{0,1\}^n$ satisfies $\boldsymbol{c}_{i-1} \preceq \hat{\boldsymbol{c}} \prec \boldsymbol{c}_i$ if and only if the first j_i digits of $\hat{\boldsymbol{c}}$ are the same as those of $\boldsymbol{c}_{i-1}$, i.e., $\hat{\boldsymbol{c}}_1^{j_i} = \boldsymbol{c}_{i-1,1}^{j_i}$. Given the length and the first j_i digits of $\hat{\boldsymbol{c}}$, the number of possible $\hat{\boldsymbol{c}}$ can be calculated based on the matrix $\boldsymbol{X}$ in the following way. Since the (β, p)-WWL constraint is local, if $j_i > \beta - 1$, the task is equivalent to calculating the number of $\tilde{\boldsymbol{c}}$ with length $n - j_i + \beta - 1$ such that the first $\beta - 1$ digits are a prefix of $\hat{\boldsymbol{c}}$, in particular, $\tilde{\boldsymbol{c}}_1^{\beta-1} = \hat{\boldsymbol{c}}_{j_i-\beta+2}^{j_i}$; otherwise, for $j_i \leqslant \beta - 1$, it is equivalent to calculating the number of $\tilde{\boldsymbol{c}}$ with length $n - j_i + \beta - 1$ such that the first $\beta - 1$ digits are zeros followed by the length-j_i prefix of $\hat{\boldsymbol{c}}$, that is, $\tilde{\boldsymbol{c}}_1^{\beta-1} = (\boldsymbol{0}_{\beta-1-j_i}, \hat{\boldsymbol{c}}_1^{j_i})$. Lines 10–15 in Algorithm 1 find the first $\beta - 1$ digits of $\tilde{\boldsymbol{c}}$ and Lines 16 and 17 calculate the number of $\tilde{\boldsymbol{c}}$, which is the number of vectors $\hat{\boldsymbol{c}}$ satisfying $\boldsymbol{c}_{i-1} \preceq \hat{\boldsymbol{c}} \prec \boldsymbol{c}_i$. Therefore, Algorithm 1 calculates $N(\boldsymbol{c}_{i-1}, \boldsymbol{c}_i)$ for $i \in \{1, \ldots, L\}$ and sums them up to derive the order of $\boldsymbol{c}$.

Time Complexity Analysis: It can be seen from the algorithm that the decoder scans the vector that is to be decoded only once. Whenever the decoder detects a 1, it uses binary searches to find the corresponding prefix vector $\boldsymbol{d}$ in $\boldsymbol{X}$, while the number of 1's is no more than $\frac{np}{\beta}$. Therefore, the time complexity of the decoder is no more that $O(\frac{np}{\beta} \log M_{\beta-1}) = O(\frac{np}{\beta} \log \sum_{i=0}^{p} \binom{\beta-1}{i}) = O(n)$, where β and p are fixed integers and not related to n.

Space Complexity Analysis: The space complexity comes from the matrix $\boldsymbol{X}$ with $n + \beta - 1$ rows and $M_{\beta-1}$ columns. Therefore, the space complexity is also $O(n)$ since β and $M_{\beta-1}$ are both fixed integers. ■

2) Encoder: The encoder follows a similar approach to map an integer $m \in \{1, \ldots, M_n\}$ to a vector $\boldsymbol{c} \in \mathcal{S}_n(\beta, p)$, such that $ord(\boldsymbol{c}) = m$. We call $\boldsymbol{c}$ the *encoded vector* for the message m. Note that $\forall m_i, m_j \in \{1, \ldots, M_n\}$, $m_i \leqslant m_j$ if and only if $\boldsymbol{c}_i \preceq \boldsymbol{c}_j$, where $ord(\boldsymbol{c}_i) = m_i$ and $ord(\boldsymbol{c}_j) = m_j$. The following encoding algorithm uses the matrix $\boldsymbol{X}$ to efficiently calculate the vector $\boldsymbol{c} \in \mathcal{S}_n(\beta, p)$ such that $ord(\boldsymbol{c}) = m$, for $m \in \{1, \ldots, M_n\}$. The algorithm has linear complexity.

Algorithm 2 Encoding: Find $\boldsymbol{c}$ such that $ord(\boldsymbol{c}) = m$

```
let cnt = 0, c = (0,...,0) with length n;
for i = 1,2,...,n{
  let t = c;
  let t(i) = 1;
  if t satisfies (β,p)-WWL constraint {
  let q = (0,...,0) with length β − 1;
/*q is a vector storing β − 2 bits to the left of t(i) in t, with a 0 appended.*/
  if (i ⩾ β − 1)
    let q_1^{β−2} = t_{i−β+2}^{i−1};
  else/*i < β − 1*/
    let q_{β−i}^{β−2} = t_1^{i−1};
  find k ∈ [1 : M_{β−1}] such that s_k = q.
  let CntTry = cnt + X(n − i + β − 1, k);
  if (CntTry + 1 < m){
    let c(i) = 1;
    let cnt = CntTry;
  }
  if (CntTry + 1 = m){
    c = t;
    return c; algorithm ends;
  }
 }
}
```

Example 7: Suppose we would like to encode one of $M_n = 421$ $(\beta = 6, p = 3)$-WWL vectors of length $n = 10$. The message to be encoded is $m = 353$.

1) $\boldsymbol{c} = (0000000000)$, $i = 1$, $\boldsymbol{t} = (1000000000)$, $\boldsymbol{q} = (00000) = s_1$, so $k = 1$. Since $cnt = 0$, $CntTry = cnt + \boldsymbol{X}(n-i+\beta-1, k) = 236 < m-1$, so set $cnt = 236$.
2) $\boldsymbol{c} = (1000000000)$, $i = 2$, $\boldsymbol{t} = (1100000000)$, $\boldsymbol{q} = (00010) = s_3$, so $k = 3$. Compute $CntTry = cnt + \boldsymbol{X}(n - i + \beta - 1, k) = 236 + \boldsymbol{X}(13, 3) = 355 > m - 1$.
3) $\boldsymbol{c} = (1000000000)$, $i = 3$, $\boldsymbol{t} = (1010000000)$, $\boldsymbol{q} = (00100) = s_5$, so $k = 5$. Compute $CntTry = cnt + \boldsymbol{X}(n - i + \beta - 1, k) = 236 + \boldsymbol{X}(12, 5) = 308 < m - 1$, so set $cnt = 308$.
4) $\boldsymbol{c} = (1010000000)$, $i = 4$, $\boldsymbol{t} = (1011000000)$, $\boldsymbol{q} = (01010) = s_{11}$, so $k = 11$. Compute $CntTry = cnt + \boldsymbol{X}(n - i + \beta - 1, k) = 308 + \boldsymbol{X}(11, 11) = 343 < m - 1$, so set $cnt = 343$.
5) $\boldsymbol{c} = (1011000000)$, $i = 5$, $\boldsymbol{t} = (1011100000)$ does not satisfy the $(6, 3)$-WWL constraint.

6) $\boldsymbol{c} = (1011000000)$, $i = 6$, $\boldsymbol{t} = (1011010000)$ does not satisfy the $(6,3)$-WWL constraint.
7) $\boldsymbol{c} = (1011000000)$, $i = 7$, $\boldsymbol{t} = (1011001000)$, $\boldsymbol{q} = (11000) = s_{23}$, so $k = 23$. Compute $CntTry = cnt + \boldsymbol{X}(n-i+\beta-1,k) = 343 + \boldsymbol{X}(8,23) = 351 < m-1$, so set $cnt = 351$.
8) $\boldsymbol{c} = (1011001000)$, $i = 8$, $\boldsymbol{t} = (1011001100)$ does not satisfy the $(6,3)$-WWL constraint.
9) $\boldsymbol{c} = (1011001000)$, $i = 9$, $\boldsymbol{t} = (1011001010)$, $\boldsymbol{q} = (00100) = s_5$, so $k = 5$. Compute $CntTry = cnt + \boldsymbol{X}(n-i+\beta-1,k) = 351 + \boldsymbol{X}(6,5) = 353 > m-1$.
10) $\boldsymbol{c} = (1011001000)$, $i = 10$, $\boldsymbol{t} = (1011001001)$, $\boldsymbol{q} = (01000) = s_9$, so $k = 9$. Compute $CntTry = cnt + \boldsymbol{X}(n-i+\beta-1,k) = 351 + \boldsymbol{X}(5,9) = 352 = m-1$. Therefore, $\boldsymbol{c} = \boldsymbol{t} = (1011001001)$ and $ord(\boldsymbol{c}) = 353$.

Theorem 10: Algorithm 2 encodes a message $m \in \{1,\ldots,M_n\}$ to a (β,p)-WWL vector $\boldsymbol{c} \in \mathcal{S}_n(\beta,p)$ such that $ord(\boldsymbol{c}) = m$, and its time complexity and space complexity are both $O(n)$.

Proof: We first show the correctness of the algorithm and then analyze its time and space complexity.

Correctness: The proof of the correctness of the encoder is similar to the proof of the correctness of the decoder. Therefore, we omit the details.

Time Complexity Analysis: It can be seen from the algorithm that the encoder scans the vector from left to right once and tries to set each entry to 1. Whenever the encoder sets an entry to 1, it first determines whether the constraint is satisfied. This takes $O(1)$ steps since we do not have to check the entire vector but only the β bits to the left of the set entry. Then, it uses binary search to find the corresponding prefix vector in $\boldsymbol{X}$, while the number of 1's is no more than $\frac{np}{\beta}$. Therefore, the complexity of the encoder is no more that $O(\frac{np}{\beta}\log M_{\beta-1}) = O(\frac{np}{\beta}\log\sum_{i=0}^{p}\binom{\beta-1}{i}) = O(n)$, where p and β are fixed numbers.

Space Complexity Analysis: The matrix $\boldsymbol{X}$ is the primary contributor to the space complexity. As is shown in the proof of Theorem 9, the space complexity is also $O(n)$. ■

Note that Algorithm 2 and Algorithm 1 establish a one-to-one mapping between $\{1,\ldots,M_n\}$ and $\mathcal{S}_n(\beta,p)$. Therefore, the rate of the encoder is maximized. If the blocklength goes to infinity, the rate of the encoder approaches $C_{\mathrm{WWL}}(\beta,p)$.

Appendix B

In this section, we present the proof of Theorem 5. The reason for which the proof of Theorem 5 is nontrivial is the following. Suppose the cell-state vector is updated from $\boldsymbol{c}_{i-1}$ to $\boldsymbol{c}_i$ on the ith write. The encoder has full knowledge of $\boldsymbol{c}_{i-1}$ and $\boldsymbol{c}_i$ since we assume there is no noise in the updating procedure. The decoder is required to recover $\boldsymbol{c}_i + \boldsymbol{c}_{i-1}$ with full knowledge of $\boldsymbol{c}_i$ but zero knowledge of $\boldsymbol{c}_{i-1}$. This is similar to the situation encountered in memories with defects, considered in [15], where the most interesting scenario is when the defect locations are available to the encoder but not to the decoder. In general, this scenario can be modeled as a channel with states [10] where the side information on states is available only to the encoder.

Proof: First, we introduce some definitions. Recall that $\mathcal{S}_n(\beta,p)$ is defined as the set all (β,p)-WWL vectors of length n. $\mathcal{S}_n(\beta,p)$ will be written as $\mathcal{S}$ for short if no confusion about the parameters can occur. Let $V_n = \{0,1\}^n$ be the n-dimensional binary vector space.

Definition 8: For a vector $\boldsymbol{x} \in V_n$ and a set $\mathcal{S} \subset V_n$, we define $\mathcal{S} + \boldsymbol{x} = \{\boldsymbol{s} + \boldsymbol{x} | \boldsymbol{s} \in \mathcal{S}\}$ and denote it by $\mathcal{S}(\boldsymbol{x})$. We call vectors in $\mathcal{S}(\boldsymbol{x})$ *reachable* by $\boldsymbol{x}$ and we say $\mathcal{S}(\boldsymbol{x})$ is *centered* at $\boldsymbol{x}$.

For two subsets $B_1, B_2 \subset V_n$, we define $B_1 + B_2 = \{\boldsymbol{b}_1 + \boldsymbol{b}_2 | \boldsymbol{b}_1 \in B_1, \boldsymbol{b}_2 \in B_2\}$. We call a subset $B \subset V_n$ $\mathcal{S}$-*good* if

$$\mathcal{S} + B = \bigcup_{\boldsymbol{b}\in B} \mathcal{S}(\boldsymbol{b}) = V_n$$

i.e., V_n is covered by the union of translates of $\mathcal{S}$ centered at vectors in B.

Lemma 3: If $B \subset V_n$ is $\mathcal{S}$-good, then $\boldsymbol{t} + B$ is $\mathcal{S}$-good, for all $\boldsymbol{t} \in V_n$.

Lemma 4: If $B \subset V_n$ is $\mathcal{S}$-good, then for all $\boldsymbol{x} \in V_n$, there exists $\boldsymbol{b} \in B$ and $\boldsymbol{s} \in \mathcal{S}$, such that $\boldsymbol{x} + \boldsymbol{s} = \boldsymbol{b}$.

Lemma 4 guarantees that if $B \subset V_n$ is an $\mathcal{S}$-good subset, then from any cell-state vector $\boldsymbol{x}$, there exists a (β,p)-WWL vector $\boldsymbol{s}$, such that $\boldsymbol{x} + \boldsymbol{s} \in B$. We skip the proofs of Lemmas 3 and 4, referring the reader to similar results and their proofs in [5].

Lemma 5: If $G_1,\ldots,G_M$ are pairwise disjoint $\mathcal{S}$-good subsets of V_n, then there exists a $(1,\beta,p)$-constrained code of size M. In particular, if G is an $\mathcal{S}$-good (n,k) linear code, then there exists a $(1,\beta,p)$-constrained code with rate $\frac{n-k}{n}$.

Proof: If G_i is $\mathcal{S}$-good for all $i \in [1:M]$, then from Lemma 4, for any $\boldsymbol{x} \in V_n$ and $i \in [1:M]$, there exist $\boldsymbol{g}_i \in G_i$ and $\boldsymbol{s}_i \in \mathcal{S}$, such that $\boldsymbol{x} + \boldsymbol{s}_i = \boldsymbol{g}_i$. Suppose the current cell-state vector is $\boldsymbol{x}$; then, we can encode the message $i \in [1:M]$ as a vector $\mathcal{E}(i,\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{s}_i \in G_i$, for some $\boldsymbol{s}_i \in \mathcal{S}$. The decoder uses the mapping $\mathcal{D}(\boldsymbol{x}) = i$, if $\boldsymbol{x} \in G_i$, to give an estimate of $i \in [1:M]$. This yields a $(1,\beta,p)$-constrained code of size M.

If $G_1,\ldots,G_{2^{n-k}}$ represent the cosets of an $\mathcal{S}$-good (n,k) linear code G, then each coset is $\mathcal{S}$-good according to Lemma 3. The rate of the resulting $(1,\beta,p)$-constrained code is $\frac{\log_2(2^{n-k})}{n} = \frac{n-k}{n}$. ■

Now we are ready to prove Theorem 5.

Let B_j be a randomly chosen (n,j) linear code with 2^j codewords ($B_0 = \{\boldsymbol{0}\}$), and let $m_{B_j} = |V_n \setminus (B_j + \mathcal{S})|$ be the number of vectors not reachable from any vector in B_j. Let $\boldsymbol{x} \in V_n$ be a randomly chosen vector and let Q_{B_j} be the probability that $\boldsymbol{x} \notin B_j + \mathcal{S}$. Then, we have

$$m_{B_j} = 2^n Q_{B_j}.$$

The proof of the following lemma is based upon ideas discussed in [3, pp. 201–202].

Lemma 6: There exists a linear code B_j such that

$$Q_{B_j} \leqslant Q_{B_0}^{2^j}.$$

Proof: Let $B_j = \{\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{2^j}\}$ denote an (n,j) linear code. If

$$S_{B_j} = B_j + \mathcal{S}$$

then

$$Q_{B_j} = 1 - 2^{-n} N_{B_j}$$

where $N_{B_j} = |S_{B_j}|$.

Let $\boldsymbol{z} \notin B_j$ and let $B_{j+1,\boldsymbol{z}}$ be the $(n, j+1)$ linear code formed by $(\boldsymbol{z} + B_j) \cup B_j$. It can be seen that $B_{j+1,\boldsymbol{z}}$ comprises the 2^j vectors in B_j plus 2^j new vectors of the form $\boldsymbol{z} + \boldsymbol{y}$, $\boldsymbol{y} \in B_j$. Let

$$S^*_{B_j,\boldsymbol{z}} = \boldsymbol{z} + S_{B_j}.$$

It can be seen that $S^*_{B_j,\boldsymbol{z}}$ has the same cardinality as S_{B_j}. Therefore, it contains N_{B_j} vectors, too, some of which may already belong to S_{B_j}. Since $S_{B_{j+1,\boldsymbol{z}}} = S_{B_j} \cup S^*_{B_j,\boldsymbol{z}}$, we have

$$N_{B_{j+1,\boldsymbol{z}}} = 2N_{B_j} - \left|S_{B_j} \cap S^*_{B_j,\boldsymbol{z}}\right|.$$

Thus, $N_{B_{j+1,\boldsymbol{z}}}$ is maximized by choosing $\boldsymbol{z}$ that minimizes $|S_{B_j} \cap S^*_{B_j,\boldsymbol{z}}|$.

Let us now calculate the average of $|S_{B_j} \cap S^*_{B_j,\boldsymbol{z}}|$ over all $\boldsymbol{z} \in V_n$. Here, all $\boldsymbol{z} \in B_j$ are also considered since they will result in an overestimate of the average of $|S_{B_j} \cap S^*_{B_j,\boldsymbol{z}}|$. Then

$$\begin{aligned}
\sum_{\boldsymbol{z}\in V_n} |S_{B_j} \cap S^*_{B_j,\boldsymbol{z}}| &= \sum_{\boldsymbol{z}\in V_n} \sum_{\boldsymbol{x}\in S_{B_j}} \mathbf{1}_{\{\boldsymbol{x}\in S^*_{B_j,\boldsymbol{z}}\}} \\
&= \sum_{\boldsymbol{x}\in S_{B_j}} \sum_{\boldsymbol{z}\in V_n} \mathbf{1}_{\{\boldsymbol{x}\in S^*_{B_j,\boldsymbol{z}}\}} \\
&\overset{(1)}{=} \sum_{\boldsymbol{x}\in S_{B_j}} \sum_{\boldsymbol{z}\in \boldsymbol{x}+S_{B_j}} 1 \\
&\overset{(2)}{=} \sum_{\boldsymbol{x}\in S_{B_j}} N_{B_j} \\
&= N^2_{B_j}
\end{aligned}$$

where $\mathbf{1}_A$ is the indicator function of the event A, i.e., $\mathbf{1}_A = 1$ if A is true and $\mathbf{1}_A = 0$ otherwise.

Equality (1) holds since, for a fixed $\boldsymbol{x}$, if $\boldsymbol{z} \in \boldsymbol{x} + S_{B_j}$, then $\boldsymbol{x} \in S^*_{B_j,\boldsymbol{z}}$ and vice versa. Equality (2) holds since $|\boldsymbol{x} + S_{B_j}| = |S_{B_j}| = N_{B_j}$. Thus, the average value of $|S_{B_j} \cap S^*_{B_j,\boldsymbol{z}}|$ is $2^{-n} N^2_{B_j}$. Since the minimum of $|S_{B_j} \cap S^*_{B_j,\boldsymbol{z}}|$ cannot exceed this average, we conclude that there exists $\boldsymbol{z} \in V_n$, such that $|S_{B_j} \cap S^*_{B_j,\boldsymbol{z}}| \leqslant 2^{-n} N^2_{B_j}$. Then, there exists B_{j+1}, such that

$$N_{B_{j+1}} \geqslant 2N_{B_j} - 2^{-n} N^2_{B_j}.$$

Thus

$$\begin{aligned}
Q_{B_{j+1}} &= 1 - 2^{-n} N_{B_{j+1}} \\
&\leqslant 1 - 2^{-n}(2N_{B_j} - 2^{-n} N^2_{B_j}) \\
&= (1 - 2^{-n} N_{B_j})^2 \\
&= Q^2_{B_j}.
\end{aligned}$$

It follows that there exists B_j, such that $Q_{B_j} \leqslant Q^{2^j}_{B_0}$. ■

Lemma 7: If $j \geqslant n - \log|\mathcal{S}| + \log n$, then there exists B_j such that $m_{B_j} < 1$.

Proof: Note that $Q_{B_0} = 1 - 2^{-n} \cdot N_{B_0} = 1 - 2^{-n} \cdot |\mathcal{S}|$. Then, there exists B_j, such that

$$\begin{aligned}
Q_{B_j} &\leqslant Q^{2^j}_{B_0} \\
&\leqslant (1 - 2^{-n}|\mathcal{S}|)^{2^j} \\
&\leqslant (1 - 2^{-n}|\mathcal{S}|)^{2^{n-\log|\mathcal{S}|+\log n}} \\
&= (1 - 2^{-n}|\mathcal{S}|)^{2^n |\mathcal{S}|^{-1} \cdot n} \\
&< e^{-n} < 2^{-n}.
\end{aligned}$$

Then, $m_{B_j} = 2^n Q_{B_j} < 1$. ■

Since m_{B_j} is an integer and $m_{B_j} < 1$, there exists an (n, j) linear code B_j such that $m_{B_j} = 0$, i.e., an $\mathcal{S}$-good B_j exists. According to Lemma 5, there exists a sequence of $(1, \beta, p)$-constrained codes of length n and rate $R_n(1, \beta, p)$ such that

$$\begin{aligned}
\sup_n R_n(1, \beta, p) &\geqslant \lim_{n\to\infty} \frac{n - (n - \log|\mathcal{S}| + \log n)}{n} \\
&= \lim_{n\to\infty} \frac{\log|\mathcal{S}| - \log n}{n} \\
&= \lim_{n\to\infty} \frac{\log|\mathcal{S}|}{n} \\
&= C_{\mathrm{WWL}}(\beta, p).
\end{aligned}$$

We have seen in Theorem 1 that $C(1, \beta, p) \leqslant C_{\mathrm{WWL}}(\beta, p)$. This concludes the proof that $C(1, \beta, p) = C_{\mathrm{WWL}}(\beta, p)$. ■

REFERENCES

[1] R. Ahlswede and Z. Zhang, "Coding for write-efficient memory," *Inf. Comput.*, vol. 83, no. 1, pp. 80–97, Oct. 1989.

[2] N. Alon and J. Spencer, *The Probabilistic Method*. New York, NY, USA: Wiley, 1992.

[3] T. Berger, *Rate Distortion Theory*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1971.

[4] G. Burr, M. Breitwisch, M. Franceschini, D. Garetto, K. Gopalakrishnan, B. Jackson, B. Kurdi, C. Lam, L. Lastras-Montaño, A. Padilla, B. Rajendran, S. Raoux, and R. Shenoy, "Phase change memory technology," *J. Vac. Sci. Technol.*, vol. 28, no. 2, pp. 223–262, Mar. 2010.

[5] G. D. Cohen, "A nonconstructive upper bound on covering radius," *IEEE Trans. Inf. Theory*, vol. 29, no. 3, pp. 352–353, May 1983.

[6] G. D. Cohen and G. Zemor, "Write-isolated memories (WIMs)," *Discrete Mathematics*, vol. 114, no. 1–3, pp. 105–113, Apr. 1983.

[7] T. Cover, "Enumerative source encoding," *IEEE Trans. Inf. Theory*, vol. 19, no. 1, pp. 73–77, Jan. 1973.

[8] S. Datta and S. W. McLaughlin, "An enumerative method for run-length-limited codes: Permutation codes," *IEEE Trans. Inf. Theory*, vol. 45, no. 6, pp. 2199–2204, Sep. 1999.

[9] S. Datta and S. W. McLaughlin, "Optimal block codes for m-ary run-length-constrained channels," *IEEE Trans. Inf. Theory*, vol. 47, no. 5, pp. 2069–2078, Jul. 2001.

[10] A. El Gamal and Y.-H. Kim, *Network Information Theory*. Cambridge, U.K.: Cambridge Univ. Press, 2011.

[11] S. Forchhammer and T. V. Laursen, "A model for the two-dimensional no isolated bits constraint," in *Proc. IEEE Int. Symp. Inf. Theory*, Seattle, WA, USA, Jul. 2006, pp. 1189–1193.

[12] F. Freitas and W. Wickle, "Storage-class memory: The next storage system technology," *IBM J. Res. Dev.*, vol. 52, no. 4, pp. 439–447, 2008.

[13] S. Halevy, J. Chen, R. M. Roth, P. H. Siegel, and J. K. Wolf, "Improved bit-stuffing bounds on two-dimensional constraints," *IEEE Trans. Inf. Theory*, vol. 50, no. 5, pp. 824–838, May 2004.

[14] C. Heegard, "On the capacity of permanent memory," *IEEE Trans. Inf. Theory*, vol. 31, no. 1, pp. 34–42, Jan. 1985.

[15] C. Heegard and A. El Gamal, "On the capacity of computer memory with defects," *IEEE Trans. Inf. Theory*, vol. 29, no. 5, pp. 731–739, Sep. 1983.

[16] K. S. Immink, P. H. Siegel, and J. K. Wolf, "Codes for digital recorders," *IEEE Trans. Inf. Theory*, vol. 44, no. 6, pp. 2260–2299, Oct. 1998.

[17] A. Jiang, J. Bruck, and H. Li, "Constrained codes for phase-change memories," in *Proc. IEEE Inf. Theory Workshop*, Dublin, Ireland, Aug. –Sep. 2010.
[18] A. Kato and K. Zeger, "On the capacity of two-dimensional run-length constrained channels," *IEEE Trans. Inf. Theory*, vol. 45, no. 5, pp. 1527–1540, Jul. 1999.
[19] L. Lastras-Montaño, M. Franceschini, T. Mittelholzer, J. Karidis, and M. Wegman, "On the lifetime of multilevel memories," in *Proc. IEEE Int. Symp. Inf. Theory*, Seoul, Korea, Jul. 2009, pp. 1224–1228.
[20] B. H. Marcus, R. M. Roth, and P. H. Siegel, "Constrained Systems and Coding for Recording Channels," in *Handbook of Coding Theory*, V. S. Pless and W. C. Huffman, Eds. New York, NY, USA: Elsevier, 1998, ch. 20.
[21] J. Moon and B. Brickner, "Maximum transition run codes for data storage systems," *IEEE Trans. Magn.*, vol. 32, no. 5, pp. 3992–3994, Sep. 1996.
[22] Z. Nagy and K. Zeger, "Asymptotic capacity of two-dimensional channels with checkerboard constraints," *IEEE Trans. Inf. Theory*, vol. 49, no. 9, pp. 2115–2125, Sep. 2003.
[23] A. Pirovano, A. Redaelli, F. Pellizzer, F. Ottogalli, M. Tosi, D. Ielmini, A. Lacaita, and R. Bez, "Reliability study of phase-change nonvolatile memories," *IEEE Trans. Device Mater. Rel.*, vol. 4, no. 3, pp. 422–427, Sep. 2004.
[24] R. Rivest and A. Shamir, "How to reuse a write-once memory," *Inf. Control*, vol. 55, no. 1–3, pp. 1–19, Dec. 1982.
[25] R. E. Swanson and J. K. Wolf, "A new class of two-dimensional RLL recording codes," *IEEE Trans. Magn.*, vol. 28, no. 6, pp. 3407–3416, Nov. 1992.
[26] I. Tal and R. M. Roth, "Convex programming upper bounds on the capacity of 2-D constraints," *IEEE Trans. Inf. Theory*, vol. 57, no. 1, pp. 381–391, Jan. 2011.
[27] W. Weeks and R. Blahut, "The capacity and coding gain of certain checkerboard codes," *IEEE Trans. Inf. Theory*, vol. 44, no. 3, pp. 1193–1203, May 1998.
[28] E. Yaakobi, S. Kayser, P. H. Siegel, A. Vardy, and J. K. Wolf, "Codes for write-once memories," *IEEE Trans. Inf. Theory*, vol. 58, no. 9, pp. 5985–5999, Sep. 2012.
[29] E. Yaakobi, S. Kayser, P. H. Siegel, A. Vardy, and J. K. Wolf, "Efficient two-write WOM-codes," in *Proc. IEEE Inf. Theory Workshop*, Dublin, Ireland, Aug.–Sep. 2010.
[30] E. Zehavi and J. K. Wolf, "On runlength codes," *IEEE Trans. Inf. Theory*, vol. 34, no. 1, pp. 45–54, Jan. 1988.

Minghai Qin (S'11) received the B.E. degree in electronic and electrical engineering from Tsinghua University, Beijing, China, in 2009. He is currently pursuing the Ph.D. degree in electrical engineering from the Department of Electrical and Computer Engineering at the University of California, San Diego, where he is associated with the Center for Magnetic Recording Research.

Eitan Yaakobi (S'07–M'12) received the B.A. degrees in computer science and mathematics, and the M.Sc. degree in computer science from the Technion-Israel Institute of Technology, Haifa, Israel, in 2005 and 2007, respectively, and the Ph.D. degree in electrical engineering from the University of California, San Diego, in 2011.

He is currently a postdoctoral researcher in electrical engineering at the California Institute of Technology, Pasadena. His research interests include coding theory, algebraic error-correction coding, and their applications for digital data storage and in particular for nonvolatile memories.

Paul H. Siegel (M'82–SM'90–F'97) received the S.B. and Ph.D. degrees in mathematics from the Massachusetts Institute of Technology (MIT), Cambridge, in 1975 and 1979, respectively.

He held a Chaim Weizmann Postdoctoral Fellowship at the Courant Institute, New York University. He was with the IBM Research Division in San Jose, CA, from 1980 to 1995. He joined the faculty at the University of California, San Diego in July 1995, where he is currently Professor of Electrical and Computer Engineering in the Jacobs School of Engineering. He is affiliated with the Center for Magnetic Recording Research where he holds an endowed chair and served as Director from 2000 to 2011. His primary research interests lie in the areas of information theory and communications, particularly coding and modulation techniques, with applications to digital data storage and transmission.

Prof. Siegel was a member of the Board of Governors of the IEEE Information Theory Society from 1991 to 1996 and from 2009 to 2011. He was re-elected for another 3-year term in 2012. He served as Co-Guest Editor of the May 1991 Special Issue on Coding for Storage Devices of the IEEE TRANSACTIONS ON INFORMATION THEORY. He served the same Transactions as Associate Editor for Coding Techniques from 1992 to 1995, and as Editor-in-Chief from July 2001 to July 2004. He was also Co-Guest Editor of the May/September 2001 two-part issue on The Turbo Principle: From Theory to Practice of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS.

Prof. Siegel was co-recipient, with R. Karabed, of the 1992 IEEE Information Theory Society Paper Award and shared the 1993 IEEE Communications Society Leonard G. Abraham Prize Paper Award with B. H. Marcus and J. K. Wolf. With J. B. Soriaga and H. D. Pfister, he received the 2007 Best Paper Award in Signal Processing and Coding for Data Storage from the Data Storage Technical Committee of the IEEE Communications Society. He holds several patents in the area of coding and detection, and was named a Master Inventor at IBM Research in 1994. He is an IEEE Fellow and a member of the National Academy of Engineering.