

# Time-Space Constrained Codes for Phase-Change Memories

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

**Abstract**—Phase-change memory (PCM) is a promising non-volatile 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, to balance the heat both in time and space.

In this paper, we study the time-space constraint for PCM, which was originally proposed by Jiang et al. A code is called an  $(\alpha, \beta, p)$ -constrained code if for any  $\alpha$  consecutive rewrites and for any segment of  $\beta$  contiguous cells, the total rewrite cost of the  $\beta$  cells over those  $\alpha$  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 et al. Then, we generalize their construction for  $(\alpha \geq 1, \beta = 1, p = 1)$ -constrained codes and show another construction for  $(\alpha = 1, \beta \geq 1, p \geq 1)$ -constrained codes. Finally, we show that these two constructions can be used to construct codes for all values of  $\alpha$ ,  $\beta$ , and  $p$ .

## I. INTRODUCTION

Phase-change memory (PCM) devices are a promising technology for non-volatile 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 [5].

While 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 non-zero value to a smaller non-zero value, one needs to first RESET the cell to level zero, and then SET it to the new desired level [5]. 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 [13]. 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 [5], [21]. 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 [5], [21]. 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 et al. [18] studied the capacity of a Write-Efficient Memory (WEM) [2] for a cost function that is associated with the write model of phase-change memories described above.

Jiang et al. [16] have proposed codes to mitigate thermal cross-talk and heat accumulation effects in PCM. Under their thermal cross-talk 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 [16], 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 [16] 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  $(\alpha, \beta, p)$ -constrained if for any  $\alpha$  consecutive rewrites and for any segment of  $\beta$  contiguous cells, the total rewrite cost of the  $\beta$  cells over those  $\alpha$  rewrites is at most  $p$ . A specific code construction was given for the  $(\alpha \geq 1, \beta = 1, p = 1)$ -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  $(\alpha = 1, \beta \geq 1, p = 1)$ -constrained codes.

The work in [16] dealt with only a few instances of the parameters  $\alpha, \beta$  and  $p$ . In this paper, we extend the code

This work was presented in part at the *IEEE Global Telecommunications Conference (GLOBECOM 2011)*, Texas, USA, (December 2011), pp. 1-6 in the proceedings.

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 at San Diego, La Jolla, CA 92093, U.S.A. (e-mail: {mqin,eyaakobi,psiegel}@ucsd.edu).

E. Yaakobi is with the Department of Electrical Engineering, California Institute of Technology, Pasadena, CA 91125 USA, and also 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: eyaakobi@ucsd.edu).

This research was supported in part by the ISEF Foundation, the Lester Deutsch Fellowship, the University of California Lab Fees Research Program, Award No. 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.

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 two-dimensional constrained coding, we present a scheme to calculate an upper bound on the achievable rate for all values of  $\alpha, \beta$  and  $p$ . If the value of  $\alpha$  or  $\beta$  is 1 then the two-dimensional constraint becomes a one-dimensional constraint and we calculate the upper bound on the achievable rate for all values of  $p$ . This result coincides with the result in [16] for  $(\alpha \geq 1, \beta = 1, p = 1)$  and  $(\alpha = 1, \beta \geq 1, p = 1)$ . 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 two-dimensional constrained codes. In Section IV, code constructions are given. First, a trivial construction is given and we show an improvement for  $(\alpha = 1, \beta \geq 1, p \geq 1)$ -constrained codes and extend the construction in [16] of  $(\alpha \geq 1, \beta = 1, p = 1)$ -constrained codes to arbitrary  $p$ . Finally, we show how to extend the constructions for all values of  $\alpha, \beta$  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  $\mathbf{u} = (u_1, \dots, u_n) \in \{0, 1\}^n$  is rewritten to another cell-state vector  $\mathbf{v} = (v_1, \dots, v_n) \in \{0, 1\}^n$ , then the rewrite cost is defined to be the Hamming distance between  $\mathbf{u}$  and  $\mathbf{v}$ , that is

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

The Hamming weight of a vector  $\mathbf{u}$  is  $wt(\mathbf{u}) = d_H(\mathbf{u}, \mathbf{0})$ . The complement of a vector  $\mathbf{u}$  is  $\bar{\mathbf{u}} = (\bar{u}_1, \dots, \bar{u}_n)$ . For a vector  $\mathbf{x} = (x_1, \dots, x_n)$ , we define for all  $1 \leq p \leq q \leq n$ ,  $\mathbf{x}_p^q = (x_p, x_{p+1}, \dots, x_q)$ . The set  $\{i, i+1, \dots, j\}$  is denoted by  $[i : j]$  for  $i \leq j$ , and in particular,  $\{1, 2, \dots, \lfloor 2^{nR} \rfloor\}$  is denoted by  $[1 : 2^{nR}]$  for an integer  $n$  and real  $R$ .

**Definition 1.** Let  $\alpha, \beta, p$  be positive integers. A code  $\mathcal{C}$  satisfies the  $(\alpha, \beta, p)$ -constraint if for any  $\alpha$  consecutive rewrites and for any segment of  $\beta$  contiguous cells, the total rewrite cost of those  $\beta$  cells over those  $\alpha$  rewrites is at most  $p$ , and  $\mathcal{C}$  is called an  $(\alpha, \beta, p)$ -constrained code. That is, if  $\mathbf{v}_i = (v_{i,1}, \dots, v_{i,n})$ , for  $i \geq 1$ , is the cell-state vector on the  $i$ -th write, then, for all  $i \geq 1$  and  $1 \leq j \leq n - \beta + 1$ ,

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

We will specify  $(\alpha, \beta, p)$ -constrained codes by an explicit construction of their encoding and decoding maps. On the  $i$ -th write, for  $i \geq 1$ , 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  $i$ -th write of the  $(\alpha, \beta, p)$ -constrained 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 the  $(\alpha, \beta, p)$ -constrained code is defined as

$$R = \lim_{m \rightarrow \infty} \frac{\sum_{i=1}^m R_i}{m}. \quad (1)$$

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  $(\alpha, \beta, p)$ -constrained code  $\mathcal{C}$  with varying individual rates to an  $(\alpha, \beta, p)$ -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 Equation (1), which is the 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 index the write number. However, arguing as in [26], it is possible to show that these extra cells do not reduce asymptotically the rate and therefore we assume here that the encoder and decoder know the write number.

A rate  $R$  is called an  $(\alpha, \beta, p)$ -achievable rate if there exists an  $(\alpha, \beta, p)$ -constrained code  $\mathcal{C}$  such that the rate of  $\mathcal{C}$  is  $R$ . We denote by  $C_n(\alpha, \beta, p)$  the supremum of all  $(\alpha, \beta, p)$ -achievable rates while fixing the number of cells to be  $n$ . The  $(\alpha, \beta, p)$ -capacity of the  $(\alpha, \beta, p)$ -constraint is denoted by  $C(\alpha, \beta, p)$  and is defined to be

$$C(\alpha, \beta, p) = \lim_{n \rightarrow \infty} C_n(\alpha, \beta, p).$$

Our goal in this paper is to give lower and upper bounds on the  $(\alpha, \beta, p)$ -capacity,  $C(\alpha, \beta, p)$ , for all values of  $\alpha, \beta$ , and  $p$ . Clearly, if  $p \geq \alpha\beta$  then  $C(\alpha, \beta, p) = 1$ . So we assume throughout the paper that  $p < \alpha\beta$ . Lower bounds will be given by specific constrained code constructions while the upper bounds will be derived analytically using tools drawn from the theory of one- and two-dimensional constrained codes.

## III. UPPER BOUND ON THE CAPACITY

In this section, we will present upper bounds on the  $(\alpha, \beta, p)$ -capacity obtained using techniques from the analysis of two-dimensional constrained codes. There are a number of two-dimensional constraints that have been extensively studied, e.g., 2-dimensional  $(d, k)$ -runlength-limited (RLL) [17], [23], no isolated bits [12], [14], and the checkerboard constraint [20], [25]. Given a two-dimensional constraint  $S$ , its capacity is defined to be

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

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 work 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  $A = (a_{i,j})_{1 \leq i \leq m, 1 \leq j \leq n} \in \{0, 1\}^{m \times n}$  is called an  $(a, b, p)$ -array if in each sub-array of  $A$  of size  $a \times b$ , the number of 1's is at most  $p$ . That is, for all  $1 \leq i \leq m - a + 1, 1 \leq j \leq n - b + 1$ ,

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

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  $a - 1$  [25].

The connection between the capacity of the two-dimensional constraint  $C_{2D}(a, b, p)$  and the  $(\alpha, \beta, p)$ -capacity is the following.

**Theorem 1.** For all  $\alpha, \beta, p$ ,  $C(\alpha, \beta, p) \leq C_{2D}(\alpha, \beta, p)$ .

*Proof:* Let  $\mathcal{C}$  be an  $(\alpha, \beta, p)$ -constrained code of length  $n$ . For any sequence of  $m$  writes, let us denote by  $v_i$ , for  $i \geq 0$ , the cell-state vector on the  $i$ -th write, where  $v_0$  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  $j$ -th cell is changed on the  $i$ -th write. Since  $\mathcal{C}$  is an  $(\alpha, \beta, p)$ -constrained code, for all  $1 \leq i \leq m - \alpha$  and  $1 \leq j \leq n - \beta + 1$ ,

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

and therefore

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

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

Every write sequence of the code  $\mathcal{C}$  corresponds to an  $(\alpha, \beta, p)$ -array and thus the number of write sequences of length  $m$  is at most the number of  $(\alpha, \beta, p)$ -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  $i$ -th write is  $R_i$ , then the total number of distinct write sequences is  $\prod_{i=1}^m 2^{nR_i}$ . We conclude that

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

and, therefore,

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

If  $m$  goes to infinity, the rate of any  $(\alpha, \beta, p)$ -constrained code  $R$  satisfies

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

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

Theorem 1 provides a scheme to calculate an upper bound on the  $(\alpha, \beta, p)$ -capacity from an upper bound on the capacity of a two-dimensional rectangular checkerboard constraint.

Unfortunately, good upper bounds are known only for some special cases of the values of  $\alpha, \beta, p$ , and in particular, when  $p = 1$ . The checkerboard constraint has attracted considerable attention over the past 20 years and some lower and upper bounds on the capacity were given in [20], [24], [25]. For instance, some upper bounds for the square checkerboard constraint are shown in [25], from which we can conclude that  $C(2, 2, 1) \leq 0.43431$  and  $C(3, 3, 1) \leq 0.25681$ .

In the rest of this section we discuss the cases where  $\alpha = 1$  or  $\beta = 1$  corresponding to one-dimensional constraints. First, let us consider the upper bound on  $C(\alpha = 1, \beta, p)$ . We use the one-dimensional  $(d, k)$ -runlength-limited (RLL) constrained codes [27], where the number of 0's between adjacent 1's is at least  $d$  and at most  $k$ . In fact, Jiang et al. [16] showed that the capacity of the  $(\beta - 1, \infty)$ -RLL constraint is an upper bound on  $C(1, \beta, 1)$ , which is a special case of Theorem 1. The lowest curve in Fig. 1 shows the capacity of the  $(\beta - 1, \infty)$ -RLL constraint. We extend the upper bounds to arbitrary  $p \geq 1$ . First, let us generalize the definition of RLL-constrained codes.

**Definition 3.** Let  $\beta, p$  be two positive integers. A binary vector  $\mathbf{u}$  satisfies the  $(\beta, p)$ -window-weight-limited (WWL) constraint if for any  $\beta$  consecutive cells there are at most  $p$  1's and  $\mathbf{u}$  is called a  $(\beta, p)$ -WWL vector. We denote the capacity of the constraint by  $C_{WWL}(\beta, p)$ .

Note that for  $p = 1$ , the  $(\beta, 1)$ -WWL constraint is the  $(\beta - 1, \infty)$ -RLL constraint. According to Theorem 1,  $C(1, \beta, p)$  is upper bounded by the capacity of the  $(\beta, p)$ -WWL constraint,  $C_{WWL}(\beta, p)$ . Thus, we are interested in finding the capacity of this constraint. The approach is similar to the one used in [25] in order to find an upper bound on the capacity of the checkerboard constraint.

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

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

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

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

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

**Example 1.** The following illustrates the construction of the  $A_{\beta=3, p=2}$  transition matrix. Note that

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

The merge of  $s_i$  and  $s_j$  for  $i, j = 1, 2, 3, 4$  determines the matrix  $A_{3,2}$ . For example,  $f_2(s_1, s_1) = (0, 0, 0)$ ,  $a_{1,1} = 1$ ;  $f_2(s_2, s_1) = F$ ,  $a_{2,1} = 0$ ;  $f_2(s_1, s_2) = (0, 0, 1)$ ,  $a_{1,2} = 1 \neq a_{2,1}$ . This shows that the matrix is not necessarily symmetric. Finally,  $f_2(s_3, s_3) = (1, 1, 1)$ , 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  $1 \leq i, j \leq M$  there exists some  $n \geq 0$  such that  $(A^n)_{i,j} > 0$ . Note that  $n$  can be a function of  $i$  and  $j$ .

**Lemma 1.** For positive integers  $\beta, p$ , the transition matrix  $A_{p,\beta}$  is irreducible.

*Proof of Lemma 1:* In our construction of  $A_{\beta,p}$ , it is possible to show that  $(A_{\beta,p}^n)_{i,j}$  is the number of vectors of length  $n + \beta - 1$  starting in  $s_i$ , ending in  $s_j$  and satisfying  $(\beta, p)$ -WWL constraint, where  $s_i$  and  $s_j$  are defined in Definition 5. Therefore,  $A_{\beta,p}$  is irreducible if such a vector of length  $n \geq 1$  exists such that it starts with  $s_i$  and ends in  $s_j$ , for every pair of  $(i, j)$ . Clearly it exists since adding zeros between  $s_i$  and  $s_j$  gives such a valid vector. This proves the irreducibility of  $A_{\beta,p}$ . ■

The next theorem is a special case of Theorem 3.9 in [19].

**Theorem 2.** The capacity of the  $(\beta, p)$ -WWL constraint is

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

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

*Proof:* See Theorem 3.9 in [19]. ■

Fig. 1 shows  $C_{WWL}(\beta, p)$ , which is the upper bound of  $C(1, \beta, p)$ , for  $p = 1, 2, 3, 4$  respectively.

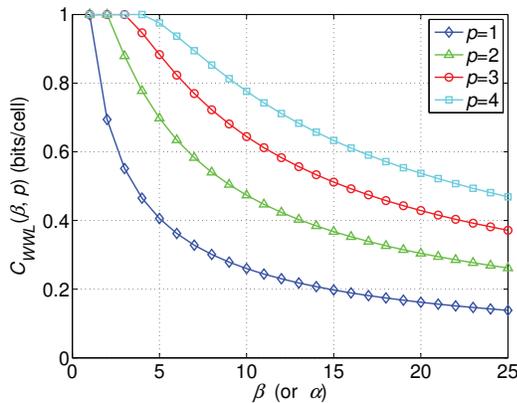


Fig. 1. Upper bound on  $C(1, \beta, p)$

**Remark 1.** There is a way of presenting the  $(\beta, p)$ -WWL constraint using labeled graphs (state transition diagrams). We

present an example of the labeled graph (transition diagram) for the  $(\beta = 7, p = 2)$ -WWL constraint in Fig. 2. An  $(\beta, p)$ -WWL vector can be generated by reading off the labels along paths in the graph and the sequences in the ellipses indicate the six most recent digits generated.

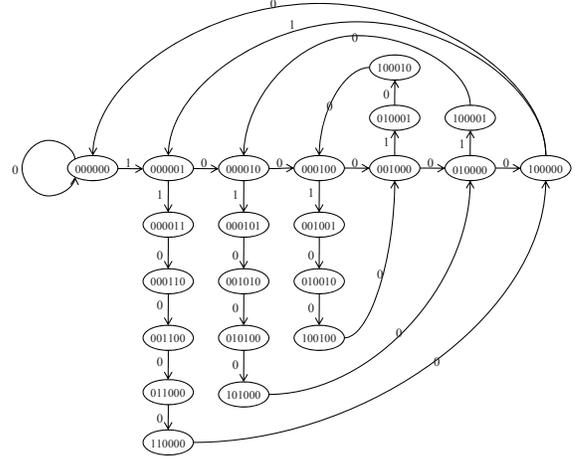


Fig. 2. Labeled graphs presenting the  $(7,2)$ -WWL constraint

**Remark 2.** According to Theorem 1, the capacity  $C(\alpha, 1, p)$  is also upper bounded by the capacity of the  $(\alpha, p)$ -WWL constraint,  $C_{WWL}(\alpha, p)$ . Jiang et al. [16] proposed an upper bound on the rate of an  $(\alpha, 1, 1)$ -constrained code with fixed block length  $n$  and multiple cell levels. By numerical experiments, we find that their upper bound appears to converge to our upper bound for binary cells when  $n \rightarrow \infty$ .

#### IV. LOWER BOUND ON THE CAPACITY

In this section, we give lower bounds on the capacity of the  $(\alpha, \beta, p)$ -constraint based upon specific code constructions. The first construction we give is a trivial one which achieves rate  $\frac{p}{\alpha\beta}$ . Then, we will show how to improve it for the cases  $(1, \beta, p)$  and  $(\alpha, 1, p)$ . In this section we assume that for all positive integers  $x, y$  the value of  $x \pmod{y}$  belongs to the group  $\{1, \dots, y\}$  via the correspondence  $0 \rightarrow y$ .

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

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

- If  $1 \leq i \pmod{\alpha} < q$ ,  $n$  bits are written to the  $n$  cells.

- If  $i \pmod{\alpha} = q$ ,  $rn/\beta$  bits are written in all cells  $c_j$  such that  $1 \leq j \pmod{\beta} \leq r$ .
- 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  $i$ -th row corresponds to the cell-state vector before the  $i$ -th write. The cells in the box in the  $i$ -th row are the only cells that can be programmed on the  $i$ -th 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  $\frac{2}{9}$ .

0:	000	000	000	000	000	000
1:	110	010	100	000	010	010
2:	110	010	100	000	010	010
3:	110	010	100	000	010	010
4:	100	100	110	010	110	110
5:	100	100	110	010	110	110
6:	100	100	110	010	110	110
7:	000	010	100	110	010	010
8:	000	010	100	110	010	010

Fig. 3. A sequence of writes of a  $(3, 3, 2)$ -constrained code

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

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

$$(x-1) \cdot \beta + y = \left( \left\lceil \frac{p}{\beta} \right\rceil - 1 \right) \beta + (p \bmod \beta) = p.$$

The total number of bits written on these  $\alpha$  writes is  $pn/\beta$  and hence the rate of the code is

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

#### A. Space Constraint Improvement

In this subsection, we improve the lower bound on  $C(1, \beta, p)$  over that offered by the trivial construction. Let  $\mathcal{S}_n(\beta, p)$  be the set of all  $(\beta, p)$ -WWL vectors of length  $n$ . We define a  $(\beta, p)$ -WWL code  $\mathcal{C}_{WWL}$  of length  $n$  as a subset of  $\mathcal{S}_n(\beta, p)$ . If the size of the code  $\mathcal{C}_{WWL}$  is  $M$ , then it is specified by an

encoding map  $\mathcal{E}_{WWL} : \{1, \dots, M\} \mapsto \mathcal{C}_{WWL}$  and a decoding map  $\mathcal{D}_{WWL} : \mathcal{C}_{WWL} \mapsto \{1, \dots, M\}$ , such that for all  $m \in \{1, \dots, M\}$ ,  $\mathcal{D}_{WWL}(\mathcal{E}_{WWL}(m)) = m$ .

The problem of finding  $(\beta, p)$ -WWL codes that achieve the capacity  $C_{WWL}(\beta, p)$  is of independent interest and we address it next. Cover [9] provided an enumerative scheme 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 [10], [11] proposed enumerative methods for binary  $(d, k)$ -RLL codes based on permutation codes. For  $(\beta, p)$ -WWL codes, we find enumerative encoding and decoding strategies with linear complexity enumerating all  $(\beta, p)$ -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  $p$  and  $\beta$ . The next construction uses  $(\beta, p)$ -WWL codes to construct  $(1, \beta, p)$ -constrained codes.

**Construction 2** Let  $\beta, p$  be positive integers such that  $p \leq \beta$ . Let  $\mathcal{C}_{WWL}$  be a  $(\beta, p)$ -WWL code of length  $n'$  and size  $M$ . Let  $\mathcal{E}_{WWL}$  and  $\mathcal{D}_{WWL}$  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  $\mathcal{E} : \{1, \dots, M\} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$  is defined for all  $(m, \mathbf{u}) \in \{1, \dots, M\} \times \{0, 1\}^n$  to be  $\mathcal{E}((m, \mathbf{u})) = \mathbf{v}$ , where
  - a)  $\mathbf{v}_1^{n'} = \mathbf{u}_1^{n'} + \mathcal{E}_{WWL}(m)$ .
  - b)  $\mathbf{v}_{n'+1}^{n'+\beta-1} = \mathbf{0}$ ,
  - c)  $\mathbf{v}_{n'+\beta}^n = \mathbf{u}_1^{n'}$ ,
- 2) The decoding map  $\mathcal{D} : \{0, 1\}^n \rightarrow \{1, \dots, M\}$  is defined for all  $\mathbf{u} \in \{0, 1\}^n$  to be

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

**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). 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  $(1011) \rightarrow (1101)$ .

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

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

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

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

The encoder  $\mathcal{E}$  and decoder  $\mathcal{D}$  come directly from  $\mathcal{E}_{WWL}$  and  $\mathcal{D}_{WWL}$ , 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  $\beta, p$  be two positive integers such that  $p \leq \beta$ , then*

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

Corollary 1 provides a lower bound that is achieved by practical coding schemes. In fact, following similar proofs in [4], [7], [8], we can prove the following theorem using probabilistic combinatorial tools [3].

**Theorem 5.** *Let  $\beta, p$  be positive integers such that  $\beta \geq p$ , then*

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

where  $C_{WWL}(\beta, p)$  is the capacity of the  $(\beta, p)$ -WWL constraint.

*Proof:* See Appendix B. ■

### B. Time Constraint Improvement

Jiang et al. constructed in [16] 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 [22]. 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  $[n, t; 2^{nR_1}, \dots, 2^{nR_t}]$  a  $t$ -write WOM code  $\mathcal{C}_W$  such that the number of messages that can be written to the memory on its  $i$ -th write is  $2^{nR_i}$ , and the sum-rate of the WOM code is defined to be  $R_{\text{sum}} = \sum_{i=1}^t R_i$ . The sum-capacity  $C_{\text{sum}}$  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  $i \in \{1, 2, \dots, t\}$ . Assuming

that the cell-state vector before the  $i$ -th write is  $\mathbf{c}_i$ , the encoder is a map

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

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

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

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

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

satisfies

$$\mathcal{D}_i(\mathcal{E}_i(m, \mathbf{c}_{i-1,1}^n)) = m.$$

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

It has been shown in [15] that the sum-capacity of a  $t$ -write WOM is  $C_{\text{sum}} = \log_2(t+1)$ .

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. On 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  $\alpha$  be a positive integer and let  $\mathcal{C}_W$  be an  $[n, t; 2^{nR_1}, \dots, 2^{nR_t}]$   $t$ -write WOM code. Let  $\mathcal{E}_i(m, \mathbf{v}_{i-1})$  be the  $i$ -th 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  $i \geq 1$ , let  $i' = i \pmod{2(t+\alpha)}$ , where  $1 \leq i' \leq 2(t+\alpha)$ . The cell-state vector after the  $i$ -th write is denoted by  $\mathbf{c}_i$ . On the  $i$ -th write, the encoder uses the following rules:*

- If  $i' \in [1 : t]$ , write  $M_{i'} \in [1 : 2^{nR_{i'}}]$  such that

$$\mathbf{c}_i = \mathcal{E}_{i'}(M_{i'}, \mathbf{c}_{i-1}).$$

- If  $i' = t+1$ , no information is written and the cell-state vector is changed to the all-one vector  $\mathbf{1}$ , i.e.,  $\mathbf{c}_i = \mathbf{1}$ .
- If  $i' \in [t+2 : t+\alpha]$ , no information is written and the cell-state vector is not changed.
- If  $i' \in [t+\alpha+1 : 2t+\alpha]$ , write  $M_{i'-t-\alpha} \in [1 : 2^{nR_{i'-t-\alpha}}]$  such that

$$\mathbf{c}_i = \overline{\mathcal{E}_{i'-t-\alpha}(M_{i'-t-\alpha}, \overline{\mathbf{c}_{i-1}})}.$$

- 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.,  $\mathbf{c}_i = \mathbf{0}$ .
- If  $i' \in [2t+\alpha+1 : 2(t+\alpha)]$ , no information is written and the cell-state vector is not changed.

**Remark 3.** This construction is presented differently in [16]. 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  $t+1$  writes, the cell is not programmed for  $\alpha-1$  writes. Therefore the rewrite cost of every cell among  $\alpha$  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  $\alpha = 4, \dots, 8$ .

$\alpha$	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  $p \geq 2$ . For simplicity of the construction, we will assume that  $p$  is an even integer; and the required modification for odd values of  $p$  will be immediately clear. We choose  $t \geq 1$  such that  $\alpha \geq (p-1)t$  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 bit-wise 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.

**Construction 4** *Let  $\alpha, p, t$  be positive integers such that  $\alpha \geq (p-1)t$ . Let  $\mathcal{C}_W$  be an  $[n, t; 2^{nR_1}, \dots, 2^{nR_t}]$   $t$ -write WOM code. For  $i \in [1 : t]$ , let  $\mathcal{E}_i(m, \mathbf{v}_{i-1})$  be its encoding map on the  $i$ -th 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  $i \geq 1$ , let  $i' = i \pmod{(\alpha+t)}$ ,  $i'' = i' \pmod{2t}$  where  $1 \leq i' \leq (\alpha+t)$ ,  $1 \leq i'' \leq 2t$ . The cell-state vector after the  $i$ -th write is denoted by  $\mathbf{c}_i$ . On the  $i$ -th write, the encoder uses the following rules:*

- If  $i' \in [1 : pt]$  and  $i'' \in [1 : t]$ , write  $M_{i''} \in [1 : 2^{nR_{i''}}]$  such that

$$\mathbf{c}_i = \mathcal{E}_{i''}(M_{i''}, \mathbf{c}_{i-1}).$$

- If  $i' \in [1 : pt]$  and  $i'' \in [t+1 : 2t]$ , write  $M_{i''-t} \in [1 : 2^{nR_{i''-t}}]$  such that

$$\mathbf{c}_i = \overline{\mathcal{E}_{i''-t}(M_{i''-t}, \overline{\mathbf{c}_{i-1}})}.$$

- If  $i' = pt+1$ , no information is written and the cell-state vector is changed to  $\mathbf{0}$ , i.e.,  $\mathbf{c}_i = \mathbf{0}$ .

- If  $i' \in [pt+2 : \alpha+t]$ , no information is written and the cell-state vector is not changed.

**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 that  $\mathcal{C}_W$  is sum-rate optimal, the rate of  $\mathcal{C}_{\alpha,1,p}$  is  $\frac{p \log_2(t+1)}{\alpha+t}$ . ■

**Remark 4.** In Construction 4 we required that  $\alpha \geq (p-1)t$  and, in particular,  $t \leq \lfloor \frac{\alpha}{p-1} \rfloor$ . If  $t > \lfloor \frac{\alpha}{p-1} \rfloor$ , 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  $\alpha, p$  be positive integer such that  $p \leq \alpha$ . Then,*

$$C(\alpha, 1, p) \geq \max_{t, t^* \in \mathbb{Z}_+} \left\{ \frac{p \log_2(t+1)}{\alpha+t}, \frac{\log_2(t^*+1)}{t^*}, \frac{p}{\alpha} \right\},$$

where

$$1 \leq t \leq \lfloor \frac{\alpha}{p-1} \rfloor, t^* = \lceil \frac{\alpha}{p-1} \rceil.$$

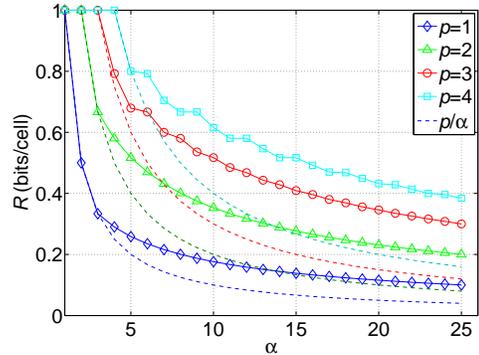


Fig. 4. Lower bound on  $C(\alpha, 1, p)$

Figure 4 shows the rates of  $(\alpha, \beta = 1, p)$  constrained codes obtained by selecting the best  $t$  for each pair of  $(\alpha, p)$ . 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  $C(\alpha, 1, p)$ .

### 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  $(\alpha, \beta, p)$ -constraints.

**Theorem 8.** For all  $\alpha, \beta, p$  positive integers,

$$C(\alpha, \beta, p) \geq \max \left\{ \frac{C(\alpha, 1, p)}{\beta}, \frac{C(1, \beta, p)}{\alpha} \right\}.$$

*Proof:* An  $(\alpha, \beta, p)$ -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  $i$ -th 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  $(\alpha, \beta, p)$ -constrained code and its rate is  $R/\alpha$ . Therefore, we conclude that  $C(\alpha, \beta, p) \geq \frac{C(1, \beta, p)}{\alpha}$ .
- 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:  $(c_1, c_2, \dots, c_{n\beta})$ . 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  $(\alpha, \beta, p)$ -constrained code and its rate is  $R/\beta$ . Therefore, we conclude that  $C(\alpha, \beta, p) \geq \frac{C(\alpha, 1, p)}{\beta}$ .

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

## APPENDIX A

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

**Definition 7.** Let  $\mathbf{X} = \{x_1, \dots, x_N\}$  be a set of distinct binary vectors,  $x_i \in \{0, 1\}^n, i = 1, \dots, N$ . Let  $\psi(\mathbf{x})$  denote the decimal representation of a vector  $\mathbf{x} \in \{0, 1\}^n$ . For  $\mathbf{x}, \mathbf{y} \in \{0, 1\}^n$ , we say  $\mathbf{x} \preceq \mathbf{y}$  (or  $\mathbf{x} \prec \mathbf{y}$ ) if and only if  $\psi(\mathbf{x}) \leq \psi(\mathbf{y})$  (or  $\psi(\mathbf{x}) < \psi(\mathbf{y})$ ). The order of the element  $x_i$  in  $\mathbf{X}$  is defined as:

$$\text{ord}(x_i) = |\{j : x_j \preceq x_i, 1 \leq j \leq N\}|.$$

Let  $\{c_1, \dots, c_{M_n}\}$  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  $(\beta, p)$ -WWL code give a one-to-one mapping between  $\mathcal{S}_n(\beta, p)$  and  $\{1, \dots, M_n\}$ , namely  $\mathcal{E}_{WWL}(m) = c_m$  where  $o(c_m) = m$  and  $\mathcal{D}_{WWL}(c_m) = o(c_m) = m$ , for all  $m = \{1, \dots, M_n\}$ . Now the problem is to calculate  $o(c_m)$  given  $c_m$ . Let  $s_1, \dots, s_{M_{\beta-1}}$  be the ordering of the vectors in  $\mathcal{S}_{\beta, p}$  introduced in Definition 5, where  $M_{\beta-1} = |\mathcal{S}_{\beta, p}| = |\mathcal{S}_{\beta-1}(\beta, p)| = \sum_{i=0}^{\beta-1} \binom{\beta-1}{i}$ . Let

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

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

**Lemma 2.** The vectors  $\mathbf{x}_{\beta, p, n+1}, n \geq \beta$ , satisfy the first-order recursion:

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

*Proof:* See [25].  $\blacksquare$

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

1) *Decoder:* Based on  $\mathbf{X}_{\beta, p, n}$ , 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  $\mathbf{c} = (c_1, \dots, c_n) \in \mathcal{S}_n(\beta, p)$  is the binary vector to be decoded; the algorithm calculates  $o(\mathbf{c}) \in \{1, \dots, M_n\}$ .

**Algorithm 1** DECODING: CALCULATE  $o(\mathbf{c}), \mathbf{c} \in \mathcal{S}_n(\beta, p)$

- 1: let  $\text{cnt} = 0, j = 1, i = 0$ ;
- 2: while  $(i \leq n)$ {
- 3:     while  $(j \leq n \text{ and } c(j) \neq 1)$
- 4:          $j = j + 1$ ;
- 5:     if  $(j = n + 1)$
- 6:          $o(\mathbf{c}) = \text{cnt} + 1$ ;
- 7:         algorithm ends;
- 8:     }
- 9:     /\*A 1 is detected in c.\*/
- 10:     let  $\mathbf{d} = (0, \dots, 0)$  with length  $\beta - 1$ ;
- 11:     /\*d is a vector storing  $\beta - 2$  bits to the left of the detected 1, appended with a 0.\*/
- 12:     if  $(j \geq \beta - 1)$
- 13:         let  $\mathbf{d}_1^{\beta-2} = \mathbf{c}_{j-\beta+2}^{j-1}$ ;
- 14:         else /\*j <  $\beta - 1$ \*/
- 15:             let  $\mathbf{d}_{\beta-j}^{\beta-2} = \mathbf{c}_1^{j-1}$ ;
- 16:             find  $k \in [1 : M_{\beta-1}]$  such that  $s_k = \mathbf{d}$ ;
- 17:              $\text{cnt} = \text{cnt} + \mathbf{X}(n - j + \beta - 1, k)$ ;
- 18:              $i = j; j = i + 1$ ;
- 19:     }
- 20:      $o(\mathbf{c}) = \text{cnt} + 1$ ;
- 21:     algorithm ends.

**Example 4.** Suppose we would like to decode a  $(6, 3)$ -WWL vector  $\mathbf{c} = (1011001001)$  of length 10.

- A 1 is detected (**1**011001001), where  $i = 0, j = 1$ . The decoder aims to find the number of vectors  $\hat{\mathbf{c}}$  such that  $(0000000000) \preceq \hat{\mathbf{c}} \prec (1000000000)$ . Now  $\mathbf{d} = (00000) = \mathbf{s}_1$ , so  $k = 1$ , and  $n - j + \beta - 1 = 14$ . Therefore,  $\text{cnt} = 0 + \mathbf{X}_{6,3,16}(14, 1) = 236$ .
- A 1 is detected (10**1**001001), where  $i = 1, j = 3$ . The decoder aims to find the number of vectors  $\hat{\mathbf{c}}$  such that  $(1000000000) \preceq \hat{\mathbf{c}} \prec (1010000000)$ . Here  $\mathbf{d} = (00100) = \mathbf{s}_5$ , so  $k = 5$ , and  $n - j + \beta - 1 = 12$ . Therefore,  $\text{cnt} = 236 + \mathbf{X}_{6,3,16}(12, 5) = 308$ .
- A 1 is detected (101**1**001001), where  $i = 3, j = 4$ . The decoder aims to find the number of vectors  $\hat{\mathbf{c}}$  such that  $(1001000000) \preceq \hat{\mathbf{c}} \prec (1011000000)$ . Here  $\mathbf{d} = (01010) = \mathbf{s}_{11}$ , so  $k = 11$ , and  $n - j + \beta - 1 = 11$ . Therefore,  $\text{cnt} = 308 + \mathbf{X}_{6,3,16}(11, 11) = 343$ .
- A 1 is detected (101100**1**001), where  $i = 4, j = 7$ . The decoder aims to find the number of vectors  $\hat{\mathbf{c}}$  such that  $(1011000000) \preceq \hat{\mathbf{c}} \prec (1011001000)$ . Here  $\mathbf{d} = (11000) = \mathbf{s}_{23}$ , so  $k = 23$ , and  $n - j + \beta - 1 = 8$ . Therefore,  $\text{cnt} = 343 + \mathbf{X}_{6,3,16}(8, 23) = 351$ .
- Finally, a 1 is detected (101100100**1**), where  $i = 7, j = 10$ . The decoder aims to find the number of vectors  $\hat{\mathbf{c}}$  such that  $(1011001000) \preceq \hat{\mathbf{c}} \prec (1011001001)$ . Here  $\mathbf{d} = (01000) = \mathbf{s}_9$ , so  $k = 9$ , and  $n - j + \beta - 1 = 5$ . Therefore,  $\text{cnt} = 351 + \mathbf{X}_{6,3,16}(5, 9) = 352$ .

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

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

*Proof:*

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

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

Let  $L$  be the number of 1's in  $\mathbf{c}$ ; let all the indices of 1's be  $j_1, j_2, \dots, j_L$  in ascending order, that is  $1 \leq j_1 < \dots < j_L \leq n$  and  $c_{j_1} = c_{j_2} = \dots = c_{j_L} = 1$ . For  $i \in \{1, \dots, L\}$ ,  $\mathbf{c}_i$  is chosen such that  $\mathbf{c}_i = \mathbf{c}_{i-1} + \delta_{j_i}$ , where  $\mathbf{c}_0 = \mathbf{0}$ , and  $\delta_j, j \in \{1, \dots, n\}$ , denotes the vector where all entries are 0 except for the  $j$ -th entry, which is a 1. Here addition is component-wise modulo-2 summation.

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

By definition, the first  $j_i - 1$  digits of  $\mathbf{c}_i$  and  $\mathbf{c}_{i-1}$  are the same, and  $c_{i,j_i} = 1$  while  $c_{i-1,j_i} = 0$ . Then a vector  $\hat{\mathbf{c}} \in \{0, 1\}^n$  satisfies  $\mathbf{c}_{i-1} \preceq \hat{\mathbf{c}} \prec \mathbf{c}_i$  if and only if the first  $j_i$  digits of  $\hat{\mathbf{c}}$  are the same as those of  $\mathbf{c}_{i-1}$ , i.e.  $\hat{\mathbf{c}}_1^{j_i} = \mathbf{c}_{i-1,1}^{j_i}$ . Given

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

*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  $\mathbf{d}$  in  $\mathbf{X}$ , while the number of 1's is no more than  $\frac{np}{\beta}$ . Therefore, the time complexity of the decoder is no more than  $O(\frac{np}{\beta} \log M_{\beta-1}) = O(\frac{np}{\beta} \log \sum_{i=0}^p \binom{\beta-1}{i}) = O(n)$ , where  $\beta$  and  $p$  are fixed integers and not related to  $n$ .

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

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

**Algorithm 2** ENCODING: FIND  $\mathbf{c}$  SUCH THAT  $o(\mathbf{c}) = m$

let  $\text{cnt} = 0, \mathbf{c} = (0, \dots, 0)$  with length  $n$ ;

for  $i = 1, 2, \dots, n$  {

let  $\mathbf{t} = \mathbf{c}$ ;

let  $t(i) = 1$ ;

if  $\mathbf{t}$  satisfies  $(\beta, p)$ -WWL constraint {

let  $\mathbf{q} = (0, \dots, 0)$  with length  $\beta - 1$ ;

*/\* $\mathbf{q}$  is a vector storing  $\beta - 2$  bits to the left of  $t(i)$  in  $\mathbf{t}$ , appended with a 0.\*/*

if  $(i \geq \beta - 1)$

let  $\mathbf{q}_1^{\beta-2} = \mathbf{t}_{i-\beta+2}^{i-1}$ ;

else */\* $i < \beta - 1$ \*/*

let  $\mathbf{q}_{\beta-i}^{\beta-2} = \mathbf{t}_1^{i-1}$ ;

find  $k \in [1 : M_{\beta-1}]$  such that  $\mathbf{s}_k = \mathbf{q}$ .

let  $\text{CntTry} = \text{cnt} + \mathbf{X}(n - i + \beta - 1, k)$ ;

if  $(\text{CntTry} + 1 = m)$  {

$\mathbf{c} = \mathbf{t}$ ;

return  $\mathbf{c}$ ; algorithm ends;

}

```

    if (CntTry + 1 < m) {
      let c(i) = 1;
      let cnt = CntTry;
    }
  }
}

```

**Example 5.** 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  $m = 353$ .

- $\mathbf{c} = (0000000000)$ ,  $i = 1$ ,  $\mathbf{t} = (1000000000)$ ,  $\mathbf{q} = (00000) = s_1$ , so  $k = 1$ . Since  $\text{cnt} = 0$ ,  $\text{CntTry} = \text{cnt} + \mathbf{X}(n - i + \beta - 1, k) = 236 < m - 1$ , so set  $\text{cnt} = 236$ .
- $\mathbf{c} = (1000000000)$ ,  $i = 2$ ,  $\mathbf{t} = (1100000000)$ ,  $\mathbf{q} = (00010) = s_3$ , so  $k = 3$ . Compute  $\text{CntTry} = \text{cnt} + \mathbf{X}(n - i + \beta - 1, k) = 236 + \mathbf{X}(13, 3) = 355 > m - 1$ .
- $\mathbf{c} = (1000000000)$ ,  $i = 3$ ,  $\mathbf{t} = (1010000000)$ ,  $\mathbf{q} = (00100) = s_5$ , so  $k = 5$ . Compute  $\text{CntTry} = \text{cnt} + \mathbf{X}(n - i + \beta - 1, k) = 236 + \mathbf{X}(12, 5) = 308 < m - 1$ , so set  $\text{cnt} = 308$ .
- $\mathbf{c} = (1010000000)$ ,  $i = 4$ ,  $\mathbf{t} = (1011000000)$ ,  $\mathbf{q} = (01010) = s_{11}$ , so  $k = 11$ . Compute  $\text{CntTry} = \text{cnt} + \mathbf{X}(n - i + \beta - 1, k) = 308 + \mathbf{X}(11, 11) = 343 < m - 1$ , so set  $\text{cnt} = 343$ .
- $\mathbf{c} = (1011000000)$ ,  $i = 5$ ,  $\mathbf{t} = (1011100000)$  does not satisfy (6, 3)-WWL constraint.
- $\mathbf{c} = (1011000000)$ ,  $i = 6$ ,  $\mathbf{t} = (1011010000)$  does not satisfy (6, 3)-WWL constraint.
- $\mathbf{c} = (1011000000)$ ,  $i = 7$ ,  $\mathbf{t} = (1011001000)$ ,  $\mathbf{q} = (11000) = s_{23}$ , so  $k = 23$ . Compute  $\text{CntTry} = \text{cnt} + \mathbf{X}(n - i + \beta - 1, k) = 343 + \mathbf{X}(8, 23) = 351 < m - 1$ , so set  $\text{cnt} = 351$ .
- $\mathbf{c} = (1011001000)$ ,  $i = 8$ ,  $\mathbf{t} = (1011001100)$  does not satisfy (6, 3)-WWL constraint.
- $\mathbf{c} = (1011001000)$ ,  $i = 9$ ,  $\mathbf{t} = (1011001010)$ ,  $\mathbf{q} = (00100) = s_5$ , so  $k = 5$ . Compute  $\text{CntTry} = \text{cnt} + \mathbf{X}(n - i + \beta - 1, k) = 351 + \mathbf{X}(6, 5) = 353 > m - 1$ .
- $\mathbf{c} = (1011001000)$ ,  $i = 10$ ,  $\mathbf{t} = (1011001001)$ ,  $\mathbf{q} = (01000) = s_9$ , so  $k = 9$ . Compute  $\text{CntTry} = \text{cnt} + \mathbf{X}(n - i + \beta - 1, k) = 351 + \mathbf{X}(5, 9) = 352 = m - 1$ . Therefore,  $\mathbf{c} = \mathbf{t} = (1011001001)$  and  $o(\mathbf{c}) = 353$ .

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

*Proof:*

*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  $\beta$  bits to the left of the set

entry. Then it uses binary search to find the corresponding prefix vector in  $\mathbf{X}$ , while the number of 1's is no more than  $\frac{np}{\beta}$ . Therefore, the complexity of the encoder is no more than  $O(\frac{np}{\beta} \log M_{\beta-1}) = O(\frac{np}{\beta} \log \sum_{i=0}^p \binom{\beta-1}{i}) = O(n)$ , where  $p$  and  $\beta$  are fixed numbers.

*Space complexity analysis:* The matrix  $\mathbf{X}$  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  $\{1, \dots, M_n\}$  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  $C_{WWL}(\beta, p)$ .

## APPENDIX B

In this section, we present the proof of Theorem 5. The reason for which the proof of Theorem 5 is non-trivial is the following. Suppose the cell-level vector is updated from  $\mathbf{c}_{i-1}$  to  $\mathbf{c}_i$  on the  $i$ -th write. The encoder has full knowledge of  $\mathbf{c}_{i-1}$  and  $\mathbf{c}_i$  since we assume there is no noise in the updating procedure. The decoder is required to recover  $\mathbf{c}_i + \mathbf{c}_{i-1}$  with full knowledge of  $\mathbf{c}_i$  but zero knowledge of  $\mathbf{c}_{i-1}$ . This is similar to the work on memories with defects in [6], where the most interesting scenario is when the defect locations are available to the encoder but not to the decoder. In general, it can be modeled as a channel with states [1] where the side information on states is only available to the encoder.

*Proof:* First we introduce some definitions. Recall that  $\mathcal{S}_n(\beta, p)$  is defined as the set all  $(\beta, p)$ -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  $\mathbf{x} \in V_n$  and a set  $\mathcal{S} \subset V_n$ , we define  $\mathcal{S} + \mathbf{x} = \{\mathbf{s} + \mathbf{x} | \mathbf{s} \in \mathcal{S}\}$  and denote it by  $\mathcal{S}(\mathbf{x})$ . We call vectors in  $\mathcal{S}(\mathbf{x})$  **reachable** by  $\mathbf{x}$  and we say  $\mathcal{S}(\mathbf{x})$  is **centered** at  $\mathbf{x}$ .

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

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

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

**Lemma 3.** If  $B \subset V_n$  is  $\mathcal{S}$ -good, then  $\mathbf{t} + B$  is  $\mathcal{S}$ -good,  $\forall \mathbf{t} \in V_n$ .

**Lemma 4.** If  $B \subset V_n$  is  $\mathcal{S}$ -good, then  $\forall \mathbf{x} \in V_n, \exists \mathbf{b} \in B, \exists \mathbf{s} \in \mathcal{S}$ , such that  $\mathbf{x} + \mathbf{s} = \mathbf{b}$ .

Lemma 4 guarantees that if  $B \subset V_n$  is an  $\mathcal{S}$ -good subset, then from any cell-state vector  $\mathbf{x}$ , there exists a  $(\beta, p)$ -WWL vector  $\mathbf{s}$ , such that  $\mathbf{x} + \mathbf{s} \in B$ . We skip the proofs of Lemma 3 and 4, referring the reader to similar results and their proofs in [7].

**Lemma 5.** If  $G_1, \dots, G_M$  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  $\mathbf{x} \in V_n$  and  $i \in [1 : M]$ , there exist  $\mathbf{g}_i \in G_i$  and  $\mathbf{s}_i \in \mathcal{S}$ , such that  $\mathbf{x} + \mathbf{s}_i = \mathbf{g}_i$ . Suppose the current cell-state vector is  $\mathbf{x}$ , then we can encode the message  $i \in [1 : M]$  as a vector  $\mathcal{E}(i, \mathbf{x}) = \mathbf{x} + \mathbf{s}_i \in G_i$ , for some  $\mathbf{s}_i \in \mathcal{S}$ . The decoder uses the mapping  $\mathcal{D}(\mathbf{x}) = i$ , if  $\mathbf{x} \in G_i$ , to give an estimate of  $i \in [1 : M]$ . This yields a  $(1, \beta, p)$ -constrained code of size  $M$ .

If  $G_1, \dots, G_{2^{n-k}}$  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 ( $B_0 = \{\mathbf{0}\}$ ), and let  $m_{B_j} = |V_n / (B_j + \mathcal{S})|$  be the number of vectors not reachable from any vector in  $B_j$ . Let  $\mathbf{x} \in V_n$  be a randomly chosen vector and let  $Q_{B_j}$  be the probability that  $\mathbf{x} \notin B_j + \mathcal{S}$ . Then we have

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

The proof of the following lemma is based upon ideas discussed in [4, pp. 201-202].

**Lemma 6.** *There exists a linear code  $B_j$  such that*

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

*Proof:* Let  $B_j = \{\mathbf{y}_1, \dots, \mathbf{y}_{2^j}\}$  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  $\mathbf{z} \notin B_j$  and let  $B_{j+1, \mathbf{z}}$  be the  $(n, j+1)$  linear code formed by  $(\mathbf{z} + B_j) \cup B_j$ . It can be seen  $B_{j+1, \mathbf{z}}$  comprises the  $2^j$  vectors in  $B_j$  plus  $2^j$  new vectors of the form  $\mathbf{z} + \mathbf{y}$ ,  $\mathbf{y} \in B_j$ . Let

$$S_{B_{j+1, \mathbf{z}}}^* = \mathbf{z} + S_{B_j}$$

It can be seen that  $S_{B_{j+1, \mathbf{z}}}^*$  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  $S_{B_{j+1, \mathbf{z}}} = S_{B_j} \cup S_{B_{j+1, \mathbf{z}}}^*$ , we have

$$N_{B_{j+1, \mathbf{z}}} = 2N_{B_j} - |S_{B_j} \cap S_{B_{j+1, \mathbf{z}}}^*|.$$

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

Let us now calculate the average of  $|S_{B_j} \cap S_{B_{j+1, \mathbf{z}}}^*|$  over all  $\mathbf{z} \in V_n$ . Here all  $\mathbf{z} \in B_j$  are also considered since they will

result in an overestimate of the average of  $|S_{B_j} \cap S_{B_{j+1, \mathbf{z}}}^*|$ . Then

$$\begin{aligned} \sum_{\mathbf{z} \in V_n} |S_{B_j} \cap S_{B_{j+1, \mathbf{z}}}^*| &= \sum_{\mathbf{z} \in V_n} \sum_{\mathbf{x} \in S_{B_j}} \mathbf{1}_{\{\mathbf{x} \in S_{B_{j+1, \mathbf{z}}}^*\}} \\ &= \sum_{\mathbf{x} \in S_{B_j}} \sum_{\mathbf{z} \in V_n} \mathbf{1}_{\{\mathbf{x} \in S_{B_{j+1, \mathbf{z}}}^*\}} \\ &\stackrel{\textcircled{1}}{=} \sum_{\mathbf{x} \in S_{B_j}} \sum_{\mathbf{z} \in \mathbf{x} + S_{B_j}} 1 \\ &\stackrel{\textcircled{2}}{=} \sum_{\mathbf{x} \in S_{B_j}} N_{B_j} \\ &= N_{B_j}^2, \end{aligned}$$

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

$$N_{B_{j+1}} \geq 2N_{B_j} - 2^n N_{B_j}^2.$$

Thus,

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

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

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

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

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

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

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

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

## REFERENCES

- [1] A. El Gamal and Y.-H. Kim, *Network Information Theory*. Cambridge: Cambridge University Press, 2011.
- [2] R. Ahlswede and Z. Zhang, "Coding for write-efficient memory," *Inform. Comput.*, vol. 83, no. 1, pp. 80 – 97, October 1989.
- [3] N. Alon and J. Spencer, *The Probabilistic Method*. New York: John Wiley Inc., 1992.
- [4] T. Berger, *Rate Distortion Theory*. Englewood Cliffs, NJ: Prentice Hall, 1971.
- [5] 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," *Journal of Vacuum Science and Technology*, vol. 28, no. 2, pp. 223–262, March 2010.
- [6] C. Heegard and A. El Gamal, "On the capacity of computer memory with defects," *IEEE Trans. Inform. Theory*, vol. 29, no. 5, pp. 731 – 739, September 1983.
- [7] G. D. Cohen, "A nonconstructive upper bound on covering radius," *Discrete Mathematics*, vol. 29, no. 3, pp. 352–353, May 1993.
- [8] G. D. Cohen and G. Zemor, "Write-isolated memories (WIMs)," *IEEE Trans. Inform. Theory*, vol. 114, no. 1-3, pp. 105–113, April 1983.
- [9] T. Cover, "Enumerative source encoding," *IEEE Trans. Inform. Theory*, vol. 19, no. 1, pp. 73 – 77, January 1973.
- [10] S. Datta and S. W. McLaughlin, "An enumerative method for runlength-limited codes: permutation codes," *IEEE Trans. Inform. Theory*, vol. 45, no. 6, pp. 2199–2204, September 1999.
- [11] —, "Optimal block codes for m-ary runlength-constrained channels," *IEEE Trans. Inform. Theory*, vol. 47, no. 5, pp. 2069–2078, July 2001.
- [12] S. Forchhammer and T. V. Laursen, "A model for the two-dimensional no isolated bits constraint," in *Proc. IEEE Int. Symp. Inform. Theory*, Seattle, Washington, July 2006, pp. 1189 – 1193.
- [13] F. Freitas and W. Wickle, "Storage-class memory: The next storage system technology," *IBM Journal of Research and Development*, vol. 52, no. 4/5, pp. 439–447, 2008.
- [14] S. Halevy, J. Chen, R. M. Roth, P. H. Siegel, and J. K. Wolf, "Improved bit-stuffing bounds on two-dimensional constraints," *IEEE Trans. Inform. Theory*, vol. 50, no. 5, pp. 824 – 838, May 2004.
- [15] C. Heegard, "On the capacity of permanent memory," *IEEE Trans. Inform. Theory*, vol. 31, no. 1, pp. 34–42, January 1985.
- [16] A. Jiang, J. Bruck, and H. Li, "Constrained codes for phase-change memories," in *Proc. IEEE Inform. Theory Workshop*, Dublin, Ireland, August-September 2010.
- [17] A. Kato and K. Zeger, "On the capacity of two-dimensional run-length constrained channels," *IEEE Trans. Inform. Theory*, vol. 45, no. 5, pp. 1527 – 1540, July 1999.
- [18] L. Lastras-Montaño, M. Franceschini, T. Mittelholzer, J. Karidis, and M. Wegman, "On the lifetime of multilevel memories," in *Proc. IEEE Int. Symp. Inform. Theory*, Seoul, Korea, July 2009, pp. 1224–1228.
- [19] B. Marcus, R. Roth, and P. H. Siegel, *Constrained Systems and Coding for Recording Channels*. Handbook of Coding Theory (V. S. Pless and W. C. Huffman, eds.), ch.20, Elsevier Science, 1998.
- [20] Z. Nagy and K. Zeger, "Asymptotic capacity of two-dimensional channels with checkerboard constraints," *IEEE Trans. Inform. Theory*, vol. 49, no. 9, pp. 2115–2125, September 2003.
- [21] 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 and Materials Reliability*, vol. 4, no. 3, pp. 422–427, September 2004.
- [22] R. Rivest and A. Shamir, "How to reuse a write-once memory," *Inform. and Contr.*, vol. 55, no. 1-3, pp. 1–19, December 1982.
- [23] R. E. Swanson and J. K. Wolf, "A new class of two-dimensional RLL recording codes," *IEEE Transactions on Magnetism*, vol. 28, no. 6, pp. 3407 – 3416, November 1992.
- [24] I. Tal and R. Roth, "Convex programming upper bounds on the capacity of 2-D constraints," *IEEE Trans. Inform. Theory*, vol. 57, no. 1, pp. 381–391, January 2011.
- [25] W. Weeks and R. Blahut, "The capacity and coding gain of certain checkerboard codes," *IEEE Trans. Inform. Theory*, vol. 44, no. 3, pp. 1193–1203, May 1998.
- [26] E. Yaakobi, S. Kayser, P. H. Siegel, A. Vardy, and J. K. Wolf, "Efficient two-write WOM-codes," in *Proc. IEEE Inform. Theory Workshop*, Dublin, Ireland, August 2010.
- [27] E. Zehavi and J. K. Wolf, "On runlength codes," *IEEE Trans. Inform. Theory*, vol. 34, no. 1, pp. 45–54, January 1988.