Compressed Rank Modulation

Qing Li
Computer Science and Engineering
Texas A & M University
College Station, TX 77840
qingli@cse.tamu.edu

***Abstract*—We propose a generalization of Rank Modulation (RM) scheme, Compressed Rank Modulation (CRM) scheme for nonvolatile memories. CRM stores information in the multiset permutation induced by the charge levels of $N = \sum_{i=1}^{n} m_i$ cells: set the ranks of the first m_1 highest charge level cells as 1, the ranks of the next m_2 highest charge level cells are 2, and the ranks of the last m_n highest charge level cells are n. The only allowed charge-placement method is the *minimal-push-up* aiming to minimize the increase of highest charge levels, and such minimization of highest charge level increase is defined as *rewriting cost*. CRM achieves a higher capacity comparing with RM, and maintains its advantages meanwhile.**

The closed-form formula for rewriting cost, sizes of balls, asymptotical rate analysis for rewriting codes, and one rewriting code construction are presented.

I. Introduction

Flash memories are nonvolatile memories both electrically programmable (charge placement) and electrically erasable (charge removal). Elements of flash memories are called cells, and traditionally data is represented by a cell charge level: let n flash memory cells denoted by $1, 2, ..., n$. $c_i \in \mathbb{R}$ denotes the charge level of cell i, if c_i is quantized to 2 discrete levels, such cell is called Single-Level Cell (SLC); if c_i is quantized to q discrete levels, it is called Multi-Level Cell (MLC).

Flash memories have the conspicuous property that the programming and the erasing are asymmetric: while adding charge to a cell is easy, removing charge has to be done with a large number ($2^{15} \sim 2^{18}$) of cells.

One problem of the MLC scheme incurred by the asymmetry is the *overshooting*: since flash memory technologies usually do not support charge removals from individual cells, the charge placement has to be done in a cautious approach to avoid the charge level exceeding the threshold charge level and the global erases, and such programming scheme speed is slow.

Another problem is the limited lifetime of flash memories: it is found that after a certain number of erasing, typically quoted at 10^4 to 10^5 depending on the specific device, the performance of flash memories becomes unreliable [2] e.g., bits in a flash chip will fail.

Rank Modulation (RM) codes [7] are proposed in the context of flash memories to eliminate mainly the overshooting problem, and they can be applied to other nonvolatile memories, such as Phase Change Memories. Besides eliminating the overshooting problem, RM has other advantages such as a faster programming speed and a better reliability.

Contrary to the MLC scheme, data of RM is represented not by the charge level but by the permutation induced by n cell charge levels. We first define following terms. For $m, n \in \mathbb{N}$, $[n]$ is the set $\{1, 2, ..., n\}$, and if $m > n$, $[n, m]$ is the set $\{n, n+1, ..., m\}$. Let $\mathcal{S}_n$ denote the set of $n!$ permutations over $[n]$. n cell charge levels, $c^n \stackrel{def}{=} (c_1, ..., c_n) \in \mathbb{R}^n$, induce a permutation $\mathbf{a} \in \mathcal{S}_n$ in the following way: the induced permutation is $\mathbf{a} \stackrel{def}{=} (a_1, a_2, ..., a_n) \in \mathcal{S}_n$ if and only if $c_{a_1} > c_{a_2} > ... > c_{a_n}$, i.e., cell a_1 has the highest charge level, and cell a_n has the lowest charge level. In this way, no discrete levels are needed (i.e., no need for threshold levels), thus it eliminates the overshooting problem.

Supposing a RM scheme represents any data of $\mathcal{D} = [D]$, the decoding function, $d : \mathcal{S}_n \to \mathcal{D}$, is to map $\mathbf{a} \in \mathcal{S}_n$ to $x \in \mathcal{D}$. The rewriting function, $r : \mathcal{S}_n \times \mathcal{D} \to \mathcal{S}_n$, is defined as given the current permutation $\mathbf{a} \in \mathcal{S}_n$ and data to rewrite $x \in \mathcal{D}$, we are seeking $\mathbf{b} = r(\mathbf{a}, x) \in \mathcal{S}_n$ such that $d(\mathbf{b}) = x$. The decoding performance is done by a charge-comparing operation to obtain the permutation and by the mapping function. The rewriting performance can be done by various ways, e.g., by a series of *push-to-the-top* [7] operations (raising the charge level of one cell above the current highest one).

In this work, we propose a generalization of RM, Compressed Rank Modulation (CRM) scheme, where the similar idea was independently presented by E. En Gad et al. [5]. For RM, in order to tolerate noise, there is a sufficiently large gap between every two analog charge levels, and thus *it constraints its capacity since every rank has one cell*. This motivates us to study CRM, where we let multiple cells share the same rank

to achieve a higher capacity, and meanwhile it maintains the advantages of RM.

Given $m^n \stackrel{def}{=} (m_1, m_2, ..., m_n) \in \mathbb{N}^n$ and $N = \sum_{i=1}^{n} m_i$, we define $\mathcal{S}_{m^n}$ as the set $\{(\mathbf{s_1}, \mathbf{s_2}, ..., \mathbf{s_n}) | \mathbf{s_i} = \{s_{i,1}, ..., s_{i,m_i}\} \subset \{1, 2, ..., N\}, \mathbf{s_i} \bigcap \mathbf{s_j} = \emptyset \text{ for } i \neq j, \text{ and } \bigcup_{i=1}^{n} \mathbf{s_i} = \{1, 2, ..., N\}\}$. When $m_1 = m_2 = ... = m_n = m$, $\mathcal{S}_{m^n}$ is denoted as $\mathcal{S}_{n,m}$. For example, $\mathcal{S}_{2,2} = \{(\{1,2\},\{3,4\}), (\{1,3\},\{2,4\}), (\{1,4\},\{2,3\}), (\{2,3\}, \{1,4\}), (\{2,4\},\{1,2\}), (\{3,4\},\{1,2\})\}$.

We define CRM as follows: given $N = \sum_{i=1}^{n} m_i$ cells, denoted as $1, 2, ..., N$, their charge levels, $c^N \stackrel{def}{=} (c_1, c_2, ..., c_N) \in \mathbb{R}^N$, and a permutation $\mathbf{a} = (\mathbf{a_1}, ..., \mathbf{a_n}) \in \mathcal{S}_{m^n}$, c^N induces $\mathbf{a}$ if and only if $c_{a_{1,j_1}} > c_{a_{2,j_2}} > ... > c_{a_{n,j_n}}$ for $j_1 \in [m_1], j_2 \in [m_2], ..., j_n \in [m_n]$, i.e., the first m_1 highest charge level cells are in $\mathbf{a}_1$, the next m_2 highest charge level cells are in $\mathbf{a}_2$, and the last m_n highest charge level cells are in $\mathbf{a_n}$. For example, let $c^4 = (3.04, 2.56, 0.98, 2.96)$ and $m^n = (2, 2)$, then the induced permutation is $(\{1,4\},\{2,3\}) \in \mathcal{S}_{2,2}$.

The decoding performance of CRM is similar to that of RM, that is though charge-comparing operations. The rewriting performance is not by the push-to-the-top operations but by the *minimal-push-up* operations, which is first presented in [4], to obtain a longer lifetime. In order to better understand it, we define *virtual level* and *rewriting cost* as follows.

As mentioned, there is no need to quantize continuous cell levels for CRM, which makes it safe for overshooting, however, also makes it hard for theoretical analysis. In order to allow easy and fair analysis, we use the virtual level concept similar to [8], which is formally defined as follows.

Given $\mathbf{u} = (\mathbf{u}_1, ..., \mathbf{u}_n) \in \mathcal{S}_{m^n}$ with $N = \sum_{i=1}^{n} m_i$, a virtual level of $\mathbf{u}$, l^N, is a vector $(l_1, ..., l_N) \in \mathbb{N}^N$ that satisfies the following conditions:

- $\forall i \in [n]$ and $j_1, j_2 \in \mathbf{u}_i$ we have $l_{j_1} = l_{j_2}$;
- $\forall 1 \leq i_1 < i_2 \leq n, j_1 \in \mathbf{u}_{i_1}$ and $j_2 \in \mathbf{u}_{i_2}$, we have $l_{j_1} > l_{j_2}$.

Besides the above two conditions, the following condition is required rewriting $\mathbf{u}$ to $\mathbf{v}$:

- Given $\mathbf{v} \in \mathcal{S}_{m^n}$ representing data to rewrite, its virtual level, $l'^N = (l'_1, ..., l'_N)$, should not decrease, i.e., $l'_i \geq l_i$ for $i \in [N]$.

The basic operation of changing $\mathbf{u}$ to $\mathbf{v}$ is a *push-up*: for cell i with rank k, $i \in \mathbf{u}_k$, the push-up operation over i is to make its charge level higher than that of all other $m_k - 1$ cells in $\mathbf{u}_k$. Our objective is *among all possible virtual levels of* $\mathbf{u}$ *and* $\mathbf{v}$*, to increase the highest virtual level from* $\mathbf{u}$ *to* $\mathbf{v}$ *as few as possible to obtain a longer lifetime*.

With this purpose, we define rewriting cost from $\mathbf{u}$ to $\mathbf{v}$, $\mathcal{C}(\mathbf{u} \to \mathbf{v})$, as $\min_{l'^N, l^N} \{\max_{i \in [N]} l'_i - \max_{i \in [N]} l_i\}$, where l'^N and l^N are virtual levels of $\mathbf{v}$ and $\mathbf{u}$, respectively. For easy and fair analysis, we assign the virtual level of $\mathbf{u}$ by letting $n, n-1, ..., 1$ to $\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_n$. The virtual level of $\mathbf{v}$ is determined as $\arg\min_{l'^N} \{\max_{i \in [N]} l'_i - n\}$. Minimal-push-up operations are the push-up operations that achieve rewriting cost.

The following example makes notions more concrete:

Example 1. Let $m^n = (3, 2, 1)$, $\mathbf{u} = (\{1,2,3\},\{4,5\},\{6\})$ and $\mathbf{v} = (\{1,3,6\},\{2,5\},\{4\})$. Let us consider virtual levels of $\mathbf{u}$ and $\mathbf{v}$ that achieve $\mathcal{C}(\mathbf{u} \to \mathbf{v})$. We assign a virtual level of $\mathbf{u}$ as $(3, 3, 3, 2, 2, 1)$, and a virtual level of $\mathbf{v}$ can be assigned as $(4, 3, 4, 2, 3, 4), (5, 3, 5, 2, 3, 5)$, which makes the highest cell rank increase from $\mathbf{u}$ to $\mathbf{v}$ 1, or others satisfying its definition, which make the highest rank increase from $\mathbf{u}$ to $\mathbf{v}$ bigger than 1. Thus, $(4, 3, 4, 2, 3, 4)$ is the one making $\mathcal{C}(\mathbf{u} \to \mathbf{v}) = 1$.

This paper is structured as follows: in section II, the programming method and rewriting cost are presented; in section III, ball sizes over $\mathcal{S}_{n,m}$ are presented; in section IV, asymptotical rate analysis of rewriting codes is presented; the conclusion is obtained in section V.

II. Rewriting data with the minimal cost

In this section, we present the way to program from $\mathbf{u}$ to $\mathbf{v}$ such that push-up operations are minimal-push-up operations, and the closed-form formula to compute rewriting cost from the given permutations directly.

A. Minimal-push-up operations

Let $\mathbf{u} = (\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_n) \in \mathcal{S}_{m^n}$ with $m^n = (m_1, m_2, ..., m_n)$ and $N = \sum_{i=1}^{n} m_i$, $i \in [n]$ and $j \in [m_i]$, $u(i,j) \stackrel{def}{=} \arg\min_k \{|\{v | v \in \mathbf{u}_i, v = 1, 2, ..., k\}| = j\}$. That is, $u(i,j)$ is the function to obtain the index of the cell, which is the j^{th} (in lexical order) among the i^{th} rank cells of $\mathbf{u}$. For example, let $u = (\{1,2,3\},\{4,5\},\{6\})$, then $u(2,2) = \arg\min_k \{|\{v | v \in \mathbf{u}_2, v = 1, 2, ..., k\}| = 2\}$, which is 5.

Reversely, let $i \in [N]$, and $u^{-1}(i)$ be the function to obtain the rank of cell i in $\mathbf{u}$ such that $i \in \mathbf{u}_{u^{-1}(i)}$.

According to the definition, a virtual level of cell $u(i,j)$, $l_{u(i,j)}$ for $i \in [n]$ and $j \in [m_i]$, of $\mathbf{u}$ can be determined as follows:

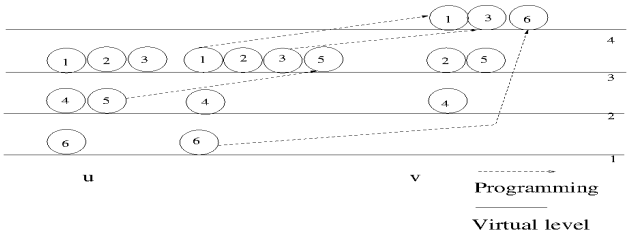

Fig. 1. An example of programming from $\mathbf{u} = (\{1,2,3\},\{4,5\},\{6\})$ to $\mathbf{v} = (\{1,3,6\},\{2,5\},\{4\})$ via minimal -push-up operations.

for $i = 1, 2, ..., n$ do:
  for $j = 1, 2, ..., m_i$ do:
    $l_{u(i,j)} \leftarrow n + 1 - i$. Then, we program $\mathbf{u}$ to $\mathbf{v}$ rank by rank, from rank n to rank 1, and an example is shown in Fig.1 to illustrate the programming process. The virtual level of $\mathbf{v}$ achieving $\mathcal{C}(\mathbf{u} \to \mathbf{v})$ can be determined through a greedy approach:

First, we identify the virtual level of the n^{th} rank cells in $\mathbf{v}$, $l_{v(n,i)}$ for $i \in [m_n]$:
for $i = 1, 2, ..., m_n$ do:
  $l_{v(n,i)} \leftarrow \max_{j \in [m_n]} l_{v(n,j)}$.

That is, the virtual level of the rank n cells in $\mathbf{v}$ is the highest virtual level of those cells of $\mathbf{u}$, which rank n in $\mathbf{v}$. For example, in Fig.1, $l_{v(3,1)} = l_4 = 2$.

Next, we identify rest cell virtual levels:
for $i = n-1, n-2, ..., 1$ do:
  for $j = 1, 2, ..., m_i$ do:
    $l_{v(i,j)} \leftarrow \max\{l_{v(i+1,j)} + 1, l_{v(i,j)}\}$.

That is, if the undecided virtual level of one cell is already higher than that of the next rank cells, which is already decided, it should stay at its original virtual level (e.g., cell 2 in Fig. 1), otherwise the virtual level has to be higher than that of the next rank cells by 1 (e.g., cell 1, 3, 5, and 6 in Fig. 1).

B. The closed-form formula for rewriting cost

According to the presented programming process, we have $\mathcal{C}(\mathbf{u} \to \mathbf{v}) = l_{v(1,i)} - n$. The next theorem presents that $\mathcal{C}(\mathbf{u} \to \mathbf{v})$ can be obtained directly from $\mathbf{u}$ and $\mathbf{v}$, instead of their virtual levels. It is actually an extension of Theorem 1 in [4], but for completeness we present its proof here:

Theorem 2. $\mathcal{C}(\mathbf{u} \to \mathbf{v}) = \max_{k \in [1,n], j \in [m_k]} (k - u^{-1}(v(k,j)))$.

Proof: First, we prove by induction on i that $l_{v(i,j)}$ is $n + 1 - i + \max_{k \in [i,n], l \in [m_k]} (k - u^{-1}(v(k,l)))$.

The base case is $i = n$. Based on the programming process, $l_{v(n,j)} = \max_{l \in [m_n]} l_{v(n,l)}$, which is $n + \max_{l \in [m_n]} (n - u^{-1}(v(n,l)))$ according to the fact that $l_{u(i,j)} = n + 1 - i$. Thus, the assumption for the base case is correct.

Now, we assume it is correct for $i = h$, and we are proving the case for $i = h - 1$.

$$\begin{aligned} l_{v(h-1,j)} &= \max\{l_{v(h,j)} + 1, l_{v(h-1,j)}\} \\ &= \max\{n + 1 - (h-1) + \max_{k \in [h,n], l \in [m_k]} \\ &\quad (k - u^{-1}(v(k,l))), n + 1 - u^{-1}(v(h-1,j))\}, \end{aligned}$$

where the first equation is based on minimal-push-up operations; the second equation is by the assumption when $i = h$ and substituting $l_{v(h-1,j)}$ by its original virtual level in $\mathbf{u}$, $n + 1 - u^{-1}(v(h-1,j))$.

Then, we proceed as:

$$\begin{aligned} &= n + 1 - (h-1) + \max\{\max_{k \in [h,n], l \in [m_k]} \\ &\quad (k - u^{-1}(v(k,l)), h - 1 - u^{-1}(v(h-1,j))\} \\ &= n + 1 - (h-1) + \max_{k \in [h-1,n], l \in [m_k]} (k - u^{-1}(v(k,l))), \end{aligned}$$

thus the conclusion about $l_{v(i,j)}$ is correct.

Therefore,

$$\mathcal{C}(\mathbf{u} \to \mathbf{v}) = l_{v(1,i)} - n = \max_{k \in [n], l \in [m_k]} (k - u^{-1}(v(k,l))).$$

■

Thus, $\mathcal{C}(\mathbf{u} \to \mathbf{v})$ is equal to the maximal rank increase of cells from $\mathbf{u}$ to $\mathbf{v}$, and $\mathcal{C}(\mathbf{u} \to \mathbf{v}) \in [n-1]$.

Given $\mathbf{u} \in \mathcal{S}_{m^n}$, $\mathbf{v} \in \mathcal{S}_{m'^n}$ with $m^n \neq m'^n$ and $N = \sum_{i=1}^{n} m_i = \sum_{i=1}^{n} m'_i$, we now generalize the above results to the rewriting case from $\mathbf{u}$ to $\mathbf{v}$, which will be used in the next section.

After applying virtual levels to $\mathbf{u}$ and $\mathbf{v}$, rewriting cost $\mathcal{C}(\mathbf{u} \to \mathbf{v})$ is still defined as $\min_{l'^N} \{\max_{i \in [N]} l'_i - n\}$ with the objective to increase the highest virtual level as few as possible, where l'^N is the virtual level for $\mathbf{v}$. Extending notations of index function as well as its inverse function to $\mathbf{u}$ and $\mathbf{v}$, we obtain that minimal-push-up operations are $l_{v(i,j)} = \max\{l_{v(i+1,j_1)} + 1, l_{v(i,j)}\}$ for $i \in [n]$, $j_1 \in [m'_{i+1}]$ and $j \in [m'_i]$. Using the skill similar to Theorem 2, we obtain that $\mathcal{C}(\mathbf{u} \to \mathbf{v}) = \max_{k \in [n], l \in [m'_k]} (k - u^{-1}(v(k,l)))$.

III. Analyzing ball sizes for rewriting codes

In this section, we present the exact number of permutations that $\forall \mathbf{u} \in \mathcal{S}_{n,m}$ can be programmed to (reps. from) $\mathbf{v} \in \mathcal{S}_{n,m}$ with rewriting cost constraint r at most.

Given $\mathbf{u} \in \mathcal{S}_{n,m}$ and $r \in [n-1]$, the outgoing ball (reps. the incoming ball) centering at $\mathbf{u}$ with radius r

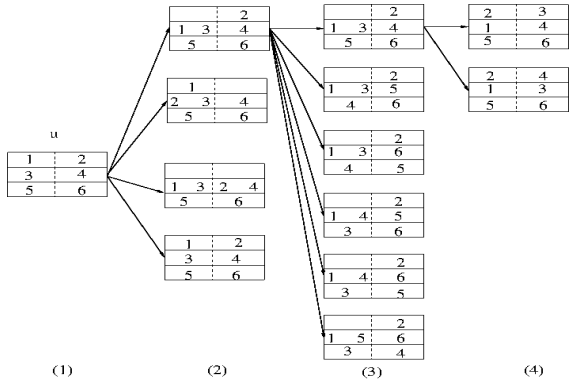

Fig. 2. An example for the outline of Theorem 3: Given $\mathbf{u} = (\{1,2\},\{3,4\},\{5,6\}) \in \mathcal{S}_{3,2}$ where cells in the same row are in the same rank, we list all $\mathbf{v} \in \mathcal{S}_{3,2}$ satisfying $\mathcal{C}(\mathbf{u} \to \mathbf{v}) \leq 1$. (a) ((1) $\to$ (2)): program some cells of rank 1 to rank 2, and there are four possible ways; (b) ((2) $\to$ (3)): cell 3, 4, 5, and 6 in $\mathbf{u}$ can be regarded as $\mathbf{u}' = (\{3,4\},\{5,6\}) \in \mathcal{S}_{2,2}$. For each cell state in (2), recursively, we program $\mathbf{u}'$ to $\mathbf{v}' \in \mathcal{S}_{2,2}$ such that $\mathcal{C}(\mathbf{u}' \to \mathbf{v}') \leq 1$ and keep the remaining cell ranks still; (c) ((3) $\to$ (4)): after (b) the cell states are not valid permutations of $\mathcal{S}_{3,2}$, e.g., $(\{2\},\{1,3,4\},\{5,6\})$, thus cells of rank 2 are programmed to the rank 1 to make it a valid permutations of $\mathcal{S}_{3,2}$, e.g., cell 3 and cell 4 in $(\{2\},\{1,3,4\},\{5,6\})$ are programmed to rank 1 forming $(\{2,3\},\{1,4\},\{5,6\})$ and $(\{2,4\},\{1,3\},\{5,6\})$, respectively.

is $\mathcal{B}^{(out)}_{n,m,r}(\mathbf{u}) = \{\mathbf{v} \in \mathcal{S}_{n,m} | \mathcal{C}(\mathbf{u} \to \mathbf{v}) \leq r\}$ (reps. $\mathcal{B}^{(in)}_{n,m,r}(\mathbf{u}) = \{\mathbf{v} \in \mathcal{S}_{n,m} | \mathcal{C}(\mathbf{v} \to \mathbf{u}) \leq r\}$).

The following theorem presents us the ball size, and it is worthy to note that E. En Gad et al. [5] derived the same result independently using the induction method.

Theorem 3. $|\mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})| = \binom{(r+1)m}{m}^{n-r} \frac{(rm)!}{m!^r}$.

Proof: We list $\mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})$ by the following steps, and we use the example of Fig.2 to illustrate it.

(a) Given $\forall (l_1, l_2, ..., l_{r+1}) \in \mathbb{N}^{r+1}$ such that $\sum_{i=1}^{r+1} l_i = m$, program l_i cells of $\mathbf{u}_1$ to rank i ($i \in [2, r+1]$) obtaining a cell state $\mathbf{a} \in \mathcal{S}_{m^n}$, where $m^n = (l_1, m + l_2, ..., m + l_{r+1}, m, ..., m)$. $\mathbf{A} \stackrel{def}{=} \{\mathbf{a}\} \subset \mathcal{S}_{m^n}$.

(b) $\forall \mathbf{a} \in \mathbf{A}$ and $\mathbf{u}_2, ..., \mathbf{u}_n$, define $\mathbf{u}' = (\mathbf{u}'_1, ..., \mathbf{u}'_{n-1}) \in \mathcal{S}_{n-1,m}$ with $\mathbf{u}'_i = \mathbf{u}_{i+1}$ ($i \in [n-1]$) as the set of permutations over $\bigcup_{i=2}^{n} \mathbf{u}_i$. Recursively, program $\mathbf{u}'$ to $\mathbf{v}' = (\mathbf{v}'_1, \mathbf{v}'_2, ..., \mathbf{v}'_{n-1}) \in \mathcal{S}_{n-1,m}$ such that $\mathcal{C}(\mathbf{u}' \to \mathbf{v}') \leq r$.
With $\mathbf{v}' \in \mathcal{S}_{n-1,m}$, the obtained cell state, $\mathbf{b} \in \mathcal{S}_{m^n}$, consists of $\mathbf{v}'$, with $b^{-1}(v'(i,j)) = i+1$ for $i \in [n-1]$ and $j \in [m]$, and $\mathbf{u}_1$ with $b^{-1}(i) = a^{-1}(i)$ for $i \in \mathbf{u}_1$, where $m^n = (l_1, m+l_2, ..., m+l_{r+1}, m, ..., m)$. $\mathbf{B} \stackrel{def}{=} \{\mathbf{b}\} \subset \mathcal{S}_{m^n}$.

(c) $\forall \mathbf{b} \in \mathbf{B}$ with its corresponding $\mathbf{v}'$ specified in (b), program l_i cells of $\mathbf{v}'_{i-1}$ for $i \in [2, r+1]$ to rank 1 obtaining a cell state $\mathbf{c} \in \mathcal{S}_{n,m}$. $\mathbf{C} \stackrel{def}{=} \{\mathbf{c}\} \subset \mathcal{S}_{n,m}$.

We are proving $\mathbf{C} = \mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})$, compute $|\mathbf{C}|$, thus $|\mathbf{C}| = |\mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})|$.

Fist, based on the outline, $\forall \mathbf{c} \in \mathbf{C}$, $\mathcal{C}(\mathbf{u} \to \mathbf{c}) \leq r$ since any cell rank of $\mathbf{c}$ is increased by r at most. Therefore $\mathbf{c} \in \mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})$, $\mathbf{C} \subseteq \mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})$ and $|\mathbf{C}| \leq |\mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})|$.

Second, we will prove $\mathcal{B}^{(out)}_{n,m,r}(\mathbf{u}) \subseteq \mathbf{C}$.

Define the following two terms:

- The set of hybrid states

The set of hybrid states of $\mathbf{u}, \mathbf{v} \in \mathcal{S}_{n,m}$ is denoted as $\mathcal{S}_h(\mathbf{u}, \mathbf{v}) \subseteq \mathcal{S}_{m^n}$, where $m^n = (m - \sum_{i=2}^{n} n_i, m + n_2, ..., m + n_n)$ and $n_i = |\{v(i,g) | v(i,g) = u(1,j) \text{ for } g, j \in [m]\}|$ for $i \in [2, n]$.

Given $\mathbf{u}$ and $\mathbf{v}$, $\forall \mathbf{m} \in \mathcal{S}_h(\mathbf{u}, \mathbf{v})$ can be specified via its ranks, $m^{-1}(i)$ for $i \in [mn]$, as follows:

for $i = 1, 2, ..., mn$ do:

if $i = u(1,j) = v(h,g)$ for $j, g \in [m]$, and $h \in [n]$, $m^{-1}(i) \leftarrow h$.

else if $i = u(h,g)$ for $h \in [n]$ and $g \in [m]$, $m^{-1}(i) \leftarrow h$.

That is, $\mathbf{m}$ is obtained by programming cells of $\mathbf{u}_1$ to their ranks in $\mathbf{v}$, and keeping the remaining cell ranks still. For example, the state of (2) in Fig. 3 is the hybrid state of $\mathbf{u}$ and $\mathbf{v}$.

Clearly, $\mathbf{m}$ is *uniquely* determined by $\mathbf{v}$ and $\mathbf{u}$, thus $|\mathcal{S}_h(\mathbf{u}, \mathbf{v})| = 1$.

Furthermore, given $\forall \mathbf{v} \in \mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})$, $\mathcal{S}_h(\mathbf{u}, \mathbf{v}) \subseteq \mathbf{A}$ since the cell ranks of $\mathbf{u_1}$ are increased by r at most.

- The set of near destination states

The set of near destination states from $\mathbf{u}' \in \mathcal{S}_{m^n}$ to $\mathbf{v} \in \mathcal{S}_{n,m}$ with $r \in [n-1]$ is denoted as $\mathcal{S}_n(\mathbf{u}', \mathbf{v}, r)$, where $m^n = (m - \sum_{i=2}^{n} n_i, m + n_2, ..., m + n_n)$ and $n_i = |\{v(i,g) | v(i,g) = u(1,j) \text{ for } g, j \in [m]\}|$ for $i \in [2, n]$.

Given $\mathbf{u}'$, $\mathbf{v}$ and r, $\forall \mathbf{m} \in \mathcal{S}_n(\mathbf{u}', \mathbf{v}, r)$ can be specified via its ranks, $m^{-1}(i)$ for $i \in [mn]$, as follows:

for $i = 1, 2, ..., mn$ do:

if $i = u'(1,h) = v(1,g)$ for $h \in [m - \sum_{i=2}^{n} n_i]$ and $g \in [m]$, $m^{-1}(i) \leftarrow 1$.

if $i = v(h,g)$ for $h \in [2, n]$ and $g \in [m]$, $m^{-1}(i) \leftarrow h$.

else if $i = u'(h,g)$ for $h \in [n]$ and $g \in [m + n_h]$, $m^{-1}(i) - h \leq r$.

That is, $\mathbf{m}$ is obtained by programming cells, which are in $\mathbf{u}'_1$ as well as in $\mathbf{v}_1$ (e.g. cell 12 in Fig. 3 if $\mathbf{u}'$ is the permutation of (2)), and cells of $\mathbf{v}_i$ for $i \in [2, n]$ (e.g. cell 10, 1, 2, 5, 11, 6, 3, 4, 7 in Fig. 3), to their ranks in $\mathbf{v}$, and increasing the remaining cells (which

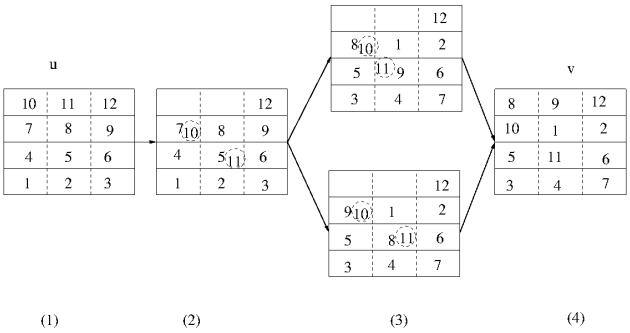

Fig. 3. An example of changing $\mathbf{u}$ to $\mathbf{v}$ in three steps, where $\mathbf{u} = (\{10,11,12\},\{7,8,9\},\{4,5,6\},\{1,2,3\})$, $\mathbf{v} = (\{8,9,12\},\{10,1,2\},\{5,11,6\},\{3,4,7\})$, and $r = \mathcal{C}(\mathbf{u} \to \mathbf{v}) = 2$. The state of (2) is the hybrid state of $\mathbf{u}$ and $\mathbf{v}$, and two states of (3) are near destination states from the state of (2) to $\mathbf{v}$ with parameter 2.

are not in $\mathbf{u'}_1$ but in $\mathbf{v}_1$, e.g. cell 8, 9 in Fig. 3) by r at most. For example, both states of (3) in Fig.3 are near destination states from $\mathbf{u}'$ to $\mathbf{v}$ with parameter 2.

Let $k_i = |\{u'(i,j)|v(1,l) = u'(i,j) \text{ for } j \in [m+n_i], l \in [m]\}|$ for $i \in [r+1]$, that is among the m cells in $\mathbf{v_1}$, the numbers of cells from $\mathbf{u'}_i$ for $i \in [r+1]$ are $k_1, k_2, ..., k_{r+1}$, respectively. Thus $|\mathcal{S}_n(\mathbf{u}', \mathbf{v}, r)| = \binom{m-k_1}{k_2,k_3,...,k_{r+1}}$. For example, in Fig. 3, let $\mathbf{u}'$ be the state of (2), $|\mathcal{S}_n(\mathbf{u}', \mathbf{v}, 2)| = 2$.

For $\forall \mathbf{w} \in \mathcal{S}_n(\mathcal{S}_h(\mathbf{u}, \mathbf{v}), \mathbf{v}, r)$, it is easy to obtain that for $i \in \mathbf{u_1}$, $w^{-1}(i) = v^{-1}(i)$, that is the cell ranks of $\mathbf{u_1}$ are the same in $\mathbf{w}$ and $\mathbf{v}$. Therefore, the permutations obtained by applying (b) on $\mathcal{S}_h(\mathbf{u}, \mathbf{v})$ must contain $\mathbf{w}$. Thus $\mathcal{S}_n(\mathcal{S}_h(\mathbf{u}, \mathbf{v}), \mathbf{v}, r) \subseteq \mathbf{B}$.

Given $\forall \mathbf{v} \in \mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})$, $\mathbf{v}$ can be obtained by first programming $\mathbf{u}$ to $\mathcal{S}_h(\mathbf{u}, \mathbf{v})$, $\mathcal{S}_h(\mathbf{u}, \mathbf{v})$ to $\forall \mathbf{w} \in \mathcal{S}_n(\mathcal{S}_h(\mathbf{u}, \mathbf{v}), \mathbf{v}, r)$, and $\mathbf{w}$ to $\mathbf{v}$. The example of Fig. 3 makes this process more concrete. Since $\mathcal{S}_h(\mathbf{u}, \mathbf{v}) \subseteq \mathbf{A}$, and $\mathcal{S}_n(\mathcal{S}_h(\mathbf{u}, \mathbf{v}), \mathbf{v}, r) \subseteq \mathbf{B}$, obtain that $\mathbf{v} \in \mathbf{C}$, $\mathcal{B}^{(out)}_{n,m,r}(\mathbf{u}) \subseteq \mathbf{C}$, and $|\mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})| \leq |\mathbf{C}|$.

Finally, we compute $|\mathbf{C}|$.

First, consider permutations without duplications.

For (a), let the numbers of cells in $\mathbf{u_1}$ programmed to the 1^{st}, the 2^{nd},..., the $(r+1)^{th}$ ranks be $l_1, l_2, ..., l_{r+1}$, respectively, thus $|\mathbf{A}| = \sum_{l_1+...+l_{r+1}=m} \binom{m}{l_1,l_2,...,l_{r+1}}$.

For (b), $|\mathbf{B}| = \sum_{l_1+...+l_{r+1}=m} \binom{m}{l_1,l_2,...,l_{r+1}} |\mathcal{B}^{(out)}_{n-1,m,r}|$.

For (c), rank i has $\binom{m}{l_i}$ ways to select and program cells to the 1^{st} rank, thus the result is

$$\sum_{l_1+...+l_{r+1}=m} \binom{m}{l_1,...,l_{r+1}} \binom{m}{l_2} ... \binom{m}{l_{r+1}} |\mathcal{B}^{(out)}_{n-1,m,r}|.$$

Next, we consider duplicated permutations in our scheme: duplications come from multiple near destination states, the number of which equals to $\binom{m-l_1}{l_2,l_3,...,l_{r+1}}$ for each combination of $l_1, l_2, ..., l_n$.

Thus, $|\mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})|$ can be written as

$$\begin{aligned}
&\sum_{l_1+...+l_{r+1}=m} \frac{\binom{m}{l_1,...,l_{r+1}}\binom{m}{l_2}...\binom{m}{l_{r+1}}}{\binom{m-l_1}{l_2,...,l_{r+1}}} |\mathcal{B}^{(out)}_{n-1,m,r}(\mathbf{u}_2..\mathbf{u}_n)| \\
= &\sum_{l_1+...+l_{r+1}=m} \binom{m}{l_1}...\binom{m}{l_{r+1}} |\mathcal{B}^{(out)}_{n-1,m,r}| \\
= &\binom{(r+1)m}{m} |\mathcal{B}^{(out)}_{n-1,m,r}| \\
= &\binom{(r+1)m}{m}^{n-r} \frac{(rm)!}{m!^r},
\end{aligned}$$

where the first equation holds by simple recursive calculations, the second equation holds by the fact that $\sum_{l_1+...+l_{r+1}=m} \binom{m}{l_1}...\binom{m}{l_{r+1}}$ equals to the coefficient of x^m in $((1+x)^m)^{r+1}$, which is $\binom{(r+1)m}{m}$, and the last one is because $|\mathcal{B}^{(out)}_{r,m,r}(...)| = |\mathcal{B}^{(out)}_{r,m,r-1}(...)| = \frac{(rm)!}{m!^r}$. ∎

Similarly, we have:

Lemma 4. $|\mathcal{B}^{(in)}_{n,m,r}(\mathbf{u})| = \binom{(r+1)m}{m}^{n-r} \frac{(rm)!}{m!^r}$.

IV. Rewriting codes with bounded cost

In this section, we study codes where the cost of the rewriting operation is limited by r. For simplicity, the analysis is mainly focus on $\mathcal{S}_{n,m}$.

A. (n, m, M, r) *rewriting codes*

Definition 5. An (n, m, M, r) rewriting code for CRM, where $r \in [n-1]$, is a collection of subsets $\mathcal{B} = \{\mathcal{B}_i | i \in [M]\}$ where $\mathcal{B}_i \subseteq \mathcal{S}_{n,m}$, and it represents data i, such that

- $\mathcal{B}_i \bigcap \mathcal{B}_j = \emptyset$ for $i \neq j$, and $\bigcup_{i=1}^{M} \mathcal{B}_i = \mathcal{S}_{n,m}$.
- $\forall \mathbf{u} \in \bigcup_{i=1}^{M} \mathcal{B}_i$, and $\forall j \in [M], \exists \mathbf{v} \in \mathcal{B}_j$ such that $\mathcal{C}(\mathbf{u} \to \mathbf{v}) \leq r$.
- $\forall \mathbf{u} \in \bigcup_{i=1}^{M} \mathcal{B}_i$, and $\forall j \in [M], \exists \mathbf{v} \in \mathcal{B}_j$ such that $\mathcal{C}(\mathbf{v} \to \mathbf{u}) \leq r$.

That is, we partition $\mathcal{S}_{n,m}$ into M disjoint sets, each set represents data, and every permutation can be programmed *to and from* some permutation of any set $\mathcal{B}_i$ for $i \in [M]$ with the cost constraint r.

Given n, m, r, the code with maximal M is called optimal. Construction 1 presents us an example of optimal code. In the following, we write a permutation of $\mathcal{S}_{3,2}$ as that of $\mathcal{S}_6$, e.g., we write $(\{1,2\},\{3,4\},\{5,6\})$ as (123456).

Construction 1. Divide the 90 codewords of $\mathcal{S}_{3,2}$ into 30 sets of 3 codewords each, where each set is a *coset* of $\langle(135)(246)\rangle$, the *cyclic group generated by* $(135)(246)$, e.g., $(123456), (345612)$ and (561234) is a *cyclic group*. Map each set to a different symbol.

Theorem 6. Code construction 1 is optimal.

Proof: The coset decomposition is a partition of $\mathcal{S}_6$. The second and the third condition can be verified easily. $|\mathcal{B}^{(in)}_{3,2,1}(\mathbf{u})| = 36$, thus each partition should be at least $\lfloor \frac{90}{36} \rfloor = 3$. Therefore the code is optimal. ∎

According to definition 5, for $\forall \mathbf{u} \in \mathcal{S}_{n,m}$ every $\mathcal{B}_i$ contains at least one element of $\mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})$ as well as that of $\mathcal{B}^{(in)}_{n,m,r}(\mathbf{u})$, thus the following corollary holds immediately:

Corollary 7. For (n, m, M, r) code, $M \leq |\mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})|$ and $M \leq |\mathcal{B}^{(in)}_{n,m,r}(\mathbf{u})|$.

B. Asymptotical rate analysis

The rate of the (n, m, M, r) code is defined as $R = \frac{\log_2 M}{nm}$, and its asymptotical rate, denoted as *storage capacity*, $\mathcal{R}(n, r)$, is defined as $\mathcal{R}(n, r) = \lim_{m\to\infty} \frac{\log_2 M}{nm}$.

Let two random variables $X, Y \in [n]$, P_{XY}, P_X and $P_{Y|X}$ denote the joint, marginal and conditional distribution, respectively. If X is uniformly distributed in the set $[n]$, we denote it as $X \sim U(1, n)$. We define a joint probability distribution set with parameters r and n, $\mathcal{P}(n, r) = \{P_{XY} | P_X = P_Y, X \sim U(1, n), P(Y|X) = 0 \text{ if } |Y - X| \geq r + 1\}$. For example, the following probability transition matrix

$$P = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{pmatrix} 2/9 & 1/9 & 0 \\ 1/9 & 1/9 & 1/9 \\ 0 & 1/9 & 2/9 \end{pmatrix} \end{array}$$

gives us an element of $\mathcal{P}(3, 1)$, where every entry p_{ij} stands for $P(X = i, Y = j)$.

The next lemma derives the characterization for $\mathcal{R}(n, r)$, and its proof uses the skill similar to [1].

Lemma 8. $\mathcal{R}(n, r) = \max_{P_{XY} \in \mathcal{P}(n,r)} H(Y|X)$.

Proof: First, we list some notations and one property of typical sequences, and for more details please refer to [3].

Let x^n be a sequence with n elements drawn from the alphabet $\mathcal{X}$. Define the type of x^n by $\pi(x|x^n) = \frac{|\{i|x_i = x\}|}{n}$. Suppose the distribution of elements in $\mathcal{X}$ is $P(\mathcal{X})$, denote it as $X \sim P(\mathcal{X})$, and the set $\mathcal{T}^n_{P_X}$ of n-sequence with type $X \sim P(\mathcal{X})$ is defined as,

$$\mathcal{T}^n_{P_X} = \{x^n | \pi(x|x^n) = P(x), \forall x\}.$$

Let (x^n, y^n) be a pair of sequences with elements drawn from alphabets $(\mathcal{X}, \mathcal{Y})$. Define their joint type: $\pi(x, y|x^n, y^n) = \frac{|\{i|(x_i, y_i) = (x, y)\}|}{n}$ for $(x, y) \in \mathcal{X} \times \mathcal{Y}$. We denote $\mathcal{T}^n_{P_{XY}}(x^n) = \{y^n | \pi(x, y|x^n, y^n) = P(x, y), \forall (x, y)\}$. The following property is useful:

$$(n + 1)^{-|\mathcal{X}||\mathcal{Y}|} 2^{nH(Y|X)} \leq |\mathcal{T}^n_{P_{XY}}(x^n)| \leq 2^{nH(Y|X)}. \tag{1}$$

For convenience, given $\forall \mathbf{u} \in \mathcal{S}_{n,m}$, below we represent it as $\mathbf{u} = (u^{-1}(1), ..., u^{-1}(mn))$.

Now, we formally prove Lemma 8 as follows:

Proof of the direct part: $\forall \mathbf{u} \in \bigcup_{i=1}^{M} \mathcal{B}_i$, we have $M \leq |\mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})|$ according to corollary 7, and thus $R \leq \frac{\log_2 |\mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})|}{mn}$.

Let Q be some joint probability distribution of $X, Y \in [n]$, and it has a marginal distribution P_X. Define $\mathcal{T}^{mn}_Q(\mathbf{u}) = \{\mathbf{v} \in \mathcal{S}_{n,m} | P_{\mathbf{u},\mathbf{v}} = Q\}$ for $\mathbf{u} \in \mathcal{S}_{n,m}$, where $P_{\mathbf{u},\mathbf{v}}$ denotes the joint type of $\mathbf{u}$ and $\mathbf{v}$.

Consider the following set of joint types $\mathcal{P}^{(out)}(\mathbf{u}, \mathbf{v}) = \{P_{\mathbf{u},\mathbf{v}} | \mathbf{v} \in \mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})\}$. Define the following set of joint types $\mathcal{P}^{(out)}(n, r) = \{P_{XY} | P_X = P_Y, X \sim U(1, n), P(Y|X) = 0 \text{ if } Y - X \geq r + 1\}$. Clearly, $\mathcal{P}(n, r) \subset \mathcal{P}^{(out)}(n, r)$.

- Now, we prove that $\mathcal{P}^{(out)}(\mathbf{u}, \mathbf{v}) = \mathcal{P}^{(out)}(n, r)$.

First, given $\mathbf{u} \in \mathcal{S}_{n,m}$ and $\forall \mathbf{v} \in \mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})$, consider $P_{\mathbf{u},\mathbf{v}}$. According to the definition of $\mathcal{S}_{n,m}$, the marginal distributions of $P_{\mathbf{u},\mathbf{v}}$, P_X, P_Y, satisfy $P_X = P_Y$ and $X \sim U(1, n)$. Based on $\mathbf{v} \in \mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})$ and Theorem 2, we conclude that $P(Y|X) = 0$ if $Y - X \geq r + 1$. Therefore, $P_{\mathbf{u},\mathbf{v}} \in \mathcal{P}^{(out)}(n, r)$ and $\mathcal{P}^{(out)}(\mathbf{u}, \mathbf{v}) \subseteq \mathcal{P}^{(out)}(n, r)$.

Next, given $\forall Q \in \mathcal{P}^{(out)}(n, r)$, and $\mathbf{u} \in \mathcal{S}_{n,m}$, consider the typical sequence with joint type Q, $\mathcal{T}^{mn}_Q(\mathbf{u})$. Based on $P_X = P_Y, X \sim U(1, n)$, we obtain that $\mathcal{T}^{mn}_Q(\mathbf{u}) \in \mathcal{S}_{n,m}$ according to the definition of $\mathcal{S}_{n,m}$. According to $P(Y|X) = 0$ if $Y - X \geq r+1$, $\mathcal{T}^{mn}_Q(\mathbf{u}) \in \mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})$ based on Theorem 2. Thus $Q \in \mathcal{P}^{(out)}(\mathbf{u}, \mathbf{v})$ and $\mathcal{P}^{(out)}(n, r) \subseteq \mathcal{P}^{(out)}(\mathbf{u}, \mathbf{v})$. □

Now, we partition $\mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})$ as follows:

$$\mathcal{B}^{(out)}_{n,m,r}(\mathbf{u}) = \bigcup_{Q \in \mathcal{P}^{(out)}(n,r)} (\mathcal{B}^{(out)}_{n,m,r}(\mathbf{u}) \cap \mathcal{T}^{mn}_Q(\mathbf{u})).$$

Since the total number of the joint types among $\mathcal{T}^{mn}_Q(\mathbf{u})$ is up bounded by $(mn + 1)^{n^2}$ [3], and the size of each joint type is up bounded by $2^{mn \max_{P' \in \mathcal{P}^{(out)}(n,r)} H(Y|X)}$ according to the right hand side inequality of (1), we obtain that

$$|\mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})| \leq (mn + 1)^{n^2} 2^{mn \max_{P' \in \mathcal{P}^{(out)}(n,r)} H(Y|X)}.$$

That is,

$$\begin{aligned} R &\leq \frac{\log_2 |\mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})|}{nm} \\ &\leq n^2 \frac{\log_2 mn+1}{mn} + \max_{P' \in \mathcal{P}^{(out)}(n,r)} H(Y|X). \end{aligned} \quad (2)$$

On the other hand, $\forall \mathbf{u} \in \bigcup_{i=1}^{M} \mathcal{B}_i$, we have $M \leq |\mathcal{B}^{(in)}_{n,m,r}(\mathbf{u})|$ based on corollary 7, thus $R \leq \frac{\log_2 |\mathcal{B}^{(in)}_{n,m,r}(\mathbf{u})|}{mn}$. Similarly, we consider the following set of joint types $\mathcal{P}^{(in)}(\mathbf{u},\mathbf{v}) = \{P_{\mathbf{u},\mathbf{v}} | \mathbf{v} \in \mathcal{B}^{(in)}_{n,m,r}(\mathbf{u})\} = \mathcal{P}^{(in)}(n,r) \stackrel{def}{=} \{P_{XY} | P_X = P_Y, X \sim U(1,n), P(Y|X) = 0 \text{ if } X - Y \geq r+1\}$. Clearly, $\mathcal{P}(n,r) \subset \mathcal{P}^{(in)}(n,r)$.

Using the same technique as above, we can obtain that

$$\begin{aligned} R &\leq \frac{\log_2 |\mathcal{B}^{(in)}_{n,m,r}(\mathbf{u})|}{nm} \\ &\leq n^2 \frac{\log_2 mn+1}{mn} + \max_{P' \in \mathcal{P}^{(in)}(n,r)} H(Y|X). \end{aligned} \quad (3)$$

By (2) and (3), we know that

$$R \leq n^2 \frac{\log_2 mn+1}{mn} + \max_{P \in \mathcal{P}(n,r)} H(Y|X).$$

Proof of the converse part: We prove this part by a random code construction.

- We first prove that random code satisfying the first two conditions of (n,m,M,r) codes does exist.

Label elements of $\mathcal{S}_{n,m}$ independently and uniformly with probability $1/M$ using one of the numbers $1,2,...,M$. The elements labelled with i form the set $\mathcal{B}_i$. M will be determined below.

We first calculate the probability $Pr(\mathbf{u},i)$ that for a fixed $\mathbf{u}$ there does not exist a $\mathbf{v} \in \mathcal{B}_i \cap \mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})$.

Clearly, $Pr(\mathbf{u},i) = (1 - \frac{1}{M})^{|\mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})|}$. Then, the probability that we do not have such random code satisfying the first two conditions of (n,m,M,r) codes is $\sum_{\mathbf{u} \in \mathcal{S}_{n,m}} \sum_{i=1}^{M} Pr(\mathbf{u},i)$, which is $\sum_{\mathbf{u} \in \mathcal{S}_{n,m}} \sum_{i=1}^{M} Pr(\mathbf{u},i) = |\mathcal{S}_{n,m}| M (1 - \frac{1}{M})^{|\mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})|} \leq 1$, if $M \leq (2\log_2 |\mathcal{S}_{n,m}|)^{-1} |\mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})|$.

With the same skill of partitioning $\mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})$, we obtain that $|\mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})|$

$$\begin{aligned} &= |\bigcup_{Q \in \mathcal{P}^{(out)}(n,r)} \mathcal{B}^{(out)}_{n,m,r}(\mathbf{u}) \bigcap \mathcal{T}^{mn}_{Q}(\mathbf{u})| \\ &\geq \max_{Q \in \mathcal{P}^{(out)}(n,r)} 2^{mnH(Y|X)} (mn+1)^{-n^2}, \end{aligned}$$

based on the left hand side inequality of (1), thus we can choose M such that

$$\frac{1}{nm} \log_2 M \leq \max_{Q \in \mathcal{P}^{(out)}(n,r)} H(Y|X) - n \frac{\log_2 mn+1}{m}, \quad (4)$$

and in this case we make sure that there exists random code satisfying the first two conditions of (n,m,M,r) codes. □

By using similar techniques, we choose M such that

$$\frac{1}{nm} \log_2 M \leq \max_{Q \in \mathcal{P}^{(in)}(n,r)} H(Y|X) - n \frac{\log_2 mn+1}{m}, \quad (5)$$

in which case we assure that there is a random code satisfying the first and the third condition of (n,m,M,r) codes.

Thus, according to (4) and (5) we choose M such that

$$\frac{1}{nm} \log_2 M \leq \max_{Q \in \mathcal{P}(n,r)} H(Y|X) - n \frac{\log_2 mn+1}{m}.$$

to make sure there is a random code satisfying all three conditions of (n,m,M,r) codes. ■

The next lemma presents us the exact value of $\mathcal{R}(n,r)$, where the matrix $A \stackrel{def}{=} [a_{i,j}]_{n \times n}$, $a_{i,j} = 1$ if $|i-j| \leq r$, and 0 otherwise, and λ_A is the largest eigenvalue of the matrix A. The proof skills are similar to those of [6].

Lemma 9. $\mathcal{R}(n,r) = \log_2 \lambda_A$.

Proof: Let (X,Y) be a pair of random variables with $P_{XY} \in \mathcal{P}(n,r)$. For $i,j \in [n]$, denote $Pr(X = i, Y = j) = q_{ij}$, $Pr(X = i) = p_i$, and $Pr(Y = j | X = i) = w_{ij}$. Then $q_{ij} = p_i \cdot w_{ij}$, and $W = [w_{ij}]_{n \times n}$ is a stochastic matrix.

According to lemma 8, $\mathcal{R}(n,r)$ can be written down as the following form:

$$\begin{aligned} \max \quad &: \quad -\sum_{i=1}^{n} \sum_{j:|j-i| \leq r} q_{ij} \log_2 (q_{ij} / \sum_{j:|j-i| \leq r} q_{ij}), \\ \text{s.t.} \quad &: \quad q_{ij} \geq 0, \text{ for } j : |j-i| \leq r \text{ and } i \in [n], \\ &\quad q_{ij} = 0, \text{ for } j : |j-i| \geq r+1 \text{ and } i \in [n], \\ &\quad \sum_{i=1}^{n} \sum_{j:|j-i| \leq r} q_{ij} = 1, \\ &\quad \sum_{i=1}^{n} q_{ij} = \sum_{i:|j-i| \leq r} q_{ji} = \frac{1}{n}, j \in [n]. \end{aligned} \quad (6)$$

We solve this via Lagrange multiplier function:

$$L(q_{ij}, j : |j-i| \leq r, i \in [n]; s; t_1, ..., t_n)$$

$$= -\sum_{i=1}^{n}\sum_{j:|j-i|\le r} q_{ij}\log_2(q_{ij}/\sum_{j:|j-i|\le r} q_{ij})$$
$$+ \quad s\sum_{i=1}^{n}\sum_{j:|j-i|\le r} q_{ij} + \sum_{j=1}^{n} t_j(\sum_{i=1}^{n} q_{ij} - \sum_{i:|j-i|\le r} q_{ji}).$$

Thus, we get $\frac{\partial L}{\partial q_{ij}} = -\log_2 w_{ij} - \mu + t_j - t_i = 0$, where $\mu = (\ln 2)^{-1}[\sum_{i=1}^{n}(|\{j : |i-j| \le r\}| - n)] - s$.

Then,

$$w_{ij} = 2^{t_j - t_i - \mu}, \text{ for } j\text{: } |j-i| \le r, \text{ and } i \in [n]. \quad (7)$$

Since $\sum_{j:|j-i|\le r} w_{ij} = 1$ for $i \in [n]$, we have

$$\sum_{j:|j-i|\le r} 2^{t_j} = 2^{\mu}2^{t_i}, \text{ for } i \in [n], \quad (8)$$

and $w_{ij} = \frac{2^{t_j}}{\sum_{l:|l-i|\le r} 2^{t_l}}$ for $j : |j-i| \le r$ and $i \in [n]$.

We interpret eq. (8) as indicating that $\{2^{t_i}\}$ is an eigenvector of the matrix $A = [a_{i,j}]_{n\times n}$, where $a_{i,j} = 1$ if$|j-i| \le r$ and 0 otherwise, thus 2^{μ} is the eigenvalue corresponding to $\{2^{t_i}\}$.

The fact that $W = [w_{ij}]_{n\times n}$ is a stochastic matrix implies that $(\frac{1}{n}, ..., \frac{1}{n})$ is its stationary distribution, which is actually p_i. This ensures that eq.(6) can be satisfied.

Substituting eq. (7) into the objective function and by simple arithmetic calculations, we obtain the desired result. ■

C. Two variants of (n, m, M, r) codes

Now, we present two variants of (n, m, M, r) codes, and we omit the proofs for the similarity to the ones of previous subsection.

Definition 10. An $(n, m, M, r)^{(in)}$ rewriting code for CRM, where $r \in [n-1]$, is a collection of subsets $\mathcal{B} = \{\mathcal{B}_i | i \in [M]\}$ where $\mathcal{B}_i \subseteq \mathcal{S}_{n,m}$, and it represents data i, such that it satisfies the first and the third condition of Definition 5.

We define its rate $R^{(in)} = \frac{\log_2 M}{nm}$, and its storage capacity $\mathcal{R}^{(in)}(n, r) = \lim_{m\to\infty} \frac{\log_2 M}{nm}$. $A^{(in)} \stackrel{def}{=} [a_{i,j}^{(in)}]_{n\times n}$, $a_{i,j}^{(in)} = 1$ if $i - j \le r$, and 0 otherwise, and $\lambda_{A^{(in)}}$ is the largest eigenvalue of the matrix $A^{(in)}$.

Lemma 11. $\mathcal{R}^{(in)}(n, r) = \max_{P_{XY}\in\mathcal{P}^{(in)}(n,r)} H(Y|X) = \log_2 \lambda_{A^{(in)}}$.

Definition 12. An $(n, m, M, r)^{(out)}$ rewriting code for CRM, where $r \in [n-1]$, is a collection of subsets $\mathcal{B} = \{\mathcal{B}_i | i \in [M]\}$ where $\mathcal{B}_i \subseteq \mathcal{S}_{n,m}$, and it represents data i, such that it satisfies the first and the second condition of Definition 5.

Similarly, we have $R^{(out)} = \frac{\log_2 M}{nm}$ and $\mathcal{R}^{(out)}(n, r) = \lim_{m\to\infty} \frac{\log_2 M}{nm}$. $A^{(out)} \stackrel{def}{=} [a_{i,j}^{(out)}]_{n\times n}$, $a_{i,j}^{(out)} = 1$ if $j - i \le r$, and 0 otherwise, and $\lambda_{A^{(out)}}$ is the largest eigenvalue of the matrix $A^{(out)}$.

Lemma 13. $\mathcal{R}^{(out)}(n, r) = \max_{P_{XY}\in\mathcal{P}^{(out)}(n,r)} H(Y|X) = \log_2 \lambda_{A^{(out)}}$.

V. Conclusion

We explore a generalized scheme of RM, CRM. General worst case code construction, and error correction codes are our future work.

VI. Acknowledgement

This work was supported in part by the NSF CAREER Award CCF-0747415 and the Israeli BSF Award 2010075. The author is especially indebted to Dr. Anxiao (Andrew) Jiang.

References

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

[2] P. Cappelletti, C. Golla, P.Olivo, and E. Zanoni, *Flash memories*, Norwell, MA: Kluwer, 1999.

[3] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: Wiley, 1991.

[4] E. En Gad, A. Jiang, and J. Bruck, "Compressed Encoding for Rank Modulation," *in Proc. 2011 IEEE Int. Symp. Information Theory*, pp. 849–853. St. Petersburg, Russia, Aug. 2011.

[5] E. En Gad, A. Jiang and J. Bruck, "Trade-offs between Instantaneous and Total Capacity in Multi-cell Flash Memories," *in Proc. 2012 IEEE Int. Symp. Information Theory*, pp. 990–994, Cambridge, MA, July 2012.

[6] F. Fu and R. W. Yeung, "On the capacity and error-correcting codes of write-efficient memories," *IEEE Trans. Inf. Theory*, vol. 46, no. 1, pp. 2299–2314, Aug. 2002.

[7] A. Jiang, R. Mateescu, M. Schwartz, and J. Bruck, "Rank Modulation for Flash memories," *IEEE Trans. Inf. Theory*, vol. 55, no.6, pp. 2659–2673, Jun. 2009.

[8] A. Jiang and Y. Wang, "Rank Modulation with Multiplicity," *in Proc. IEEE Workshop on Application of Communication Theory to Emerging Memory Technologies (ACTEMT)*, pp. 1928–1932, Miami, FL, Dec. 2010.