Rank Modulation with Multiplicity

Anxiao (Andrew) Jiang
Computer Science and Eng. Dept.
Texas A&M University
College Station, TX 77843
ajiang@cse.tamu.edu

Yue Wang
Computer Science and Eng. Dept.
Texas A&M University
College Station, TX 77843
yuewang@cse.tamu.edu

***Abstract*—Rank modulation is a scheme that uses the relative order of cell levels to represent data. Its applications include flash memories, phase-change memories, etc. An extension of rank modulation is studied in this paper, where multiple cells can have the same rank. We focus on the rewriting of data based on this new scheme, and study its basic properties.**

I. Introduction

Rank modulation is a scheme that uses the relative order of cell levels to represent data [3]. Its applications include flash memories, phase-change memories (PCMs), etc. The cell level corresponds to the cell's charge level for flash memories and the cell's electrical resistance for PCMs, and is a real number. Consider n cells $c_1, c_2, \ldots, c_n$ whose levels are $\ell_1, \ell_2, \ldots, \ell_n$, respectively, where $\ell_i \neq \ell_j$ when $i \neq j$. Let $(a_1, a_2, \ldots, a_n)$ be a permutation of the set $\{1, 2, \ldots, n\}$, such that $\ell_{a_1} > \ell_{a_2} > \cdots > \ell_{a_n}$. Then for $1 \leq i \leq n$, the cell c_{a_i} has the i-th highest level and is said to have *rank* i. The rank modulation scheme uses the ranks of cells (instead of the real values of the cell levels) to represent data; namely, the information bits are mapped to the permutation $(a_1, a_2, \ldots, a_n)$ [3].

Rank modulation can make it simpler and more robust to program flash memory cells, where the cell levels are only allowed to monotonically increase during the programming process. For PCMs, if the cells are programmed only through crystallization without the reset operation (which can happen for multi-level cells or rewriting codes), the same benefit can be obtained. It is also robust against asymmetric noise in cell levels, such as the charge-leakage noise of flash memories and the long-term cell crystallization noise of PCMs. There has been a number of works studying rank modulation[3], [5], [7], [8] and its error-correcting codes [1], [2], [4], [6].

In this paper, we study an extension of rank modulation, where multiple cells can have the same rank. The general idea is that we see cells of similar levels as having the same rank, and see cells of sufficiently different levels as having different ranks. There are naturally various ways to define the similarity of cell levels, including the following one. Let Δ and δ be two parameters, where $\Delta \geq \delta \geq 0$. For n cells whose levels can be ordered as $\ell_{a_1} \geq \ell_{a_2} \geq \cdots \geq \ell_{a_n}$, we require that for $1 \leq i < n$, either $\ell_{a_i} - \ell_{a_{i+1}} \leq \delta$ or $\ell_{a_i} - \ell_{a_{i+1}} > \Delta$. Then for $1 \leq i < n$, if $\ell_{a_i} - \ell_{a_{i+1}} \leq \delta$, we say the cells c_{a_i} and $c_{a_{i+1}}$ have the same *rank*; if $\ell_{a_i} - \ell_{a_{i+1}} > \Delta$, we say they have different *ranks*. For example, assume $\delta = 0.2$, $\Delta = 0.5$, $n = 8$ and

$$(\ell_1, \ldots, \ell_8) = (0.8,\ 2.2,\ 1.56,\ 0.21,\ 0.2,\ 2.1,\ 1.35,\ 1.38).$$

Then $(a_1, \ldots, a_8) = (2, 6, 3, 8, 7, 1, 4, 5)$, and

$$(\ell_{a_1}, \ldots, \ell_{a_8}) = (2.2,\ 2.1,\ 1.56,\ 1.38,\ 1.35,\ 0.8,\ 0.21,\ 0.2);$$

so the cells c_2, c_6 have rank 1, c_3, c_8, c_7 have rank 2, c_1 has rank 3, and c_4, c_5 have rank 4. (We may further bound the maximum difference between the levels of the cells of the same rank.) Here the parameter Δ ensures the cell levels for different ranks are sufficiently apart so that they can tolerate noise better, and δ is chosen appropriately so that the cell levels for the same rank can be programmed successfully with high probability. Allowing cells to have the same rank can help achieve higher storage capacity. And since the gap between the cell levels of different ranks does not have a specific required value – in particular it is not upper bounded – the cells can still be programmed easily without the risk of charge overshooting (as long as the cell levels of each individual rank are programmed well.) We can use the same low-rank-to-high-rank method to program cells as in [3]. Note that when $\delta = \Delta = 0$, as no two cells can practically have exactly the same level, the scheme is reduced to the original rank modulation where every cell has a distinct rank [3].

Let $\mathcal{S}_n = \{(s_1, s_2, \ldots, s_k) \mid 1 \leq k \leq n;\ s_i \subseteq \{1, 2, \ldots, n\}$ and $|s_i| \geq 1$ for $1 \leq i \leq k;\ \cup_{i=1}^{k} s_i = \{1, \ldots, n\};\ s_i \cap s_j = \emptyset$ for $i \neq j\}$. Every element $(s_1, s_2, \ldots, s_k)$ in $\mathcal{S}_n$ is a partition of the set $\{1, 2, \ldots, n\}$. We use $(s_1, s_2, \ldots, s_k)$ to denote the cells' ranks, where for $1 \leq i \leq k$, the cells with indices in s_i have the rank i. (For the previous example, we have $(s_1, s_2, \ldots, s_k) = (\{2, 6\}, \{3, 7, 8\}, \{1\}, \{4, 5\})$.) The data are represented by the elements of $\mathcal{S}_n$. Note that the difficulty of programming cells varies for the different elements of $\mathcal{S}_n$. It is simple to program two cells into different ranks since we only need the gap between their levels to be sufficiently large; but it is more challenging to program cells into the same rank because their levels need to be similar. The more cells share the same rank, the more difficult it is to program them. In the following, we consider only the elements of $\mathcal{S}_n$ where every rank accommodates at most λ cells; that is, let $\mathcal{S}_{n,\lambda} = \{(s_1, s_2, \ldots, s_k) \in \mathcal{S}_n \mid \forall\ i,\ |s_i| \leq \lambda\}$, and we use only the elements of $\mathcal{S}_{n,\lambda}$ to represent data. The parameter λ determines the tradeoff between the complexity of cell programming and the storage capacity. We call the scheme *rank modulation with multiplicity* λ.

In this paper, we focus on the rewriting of data based on the new rank modulation scheme. We study its basic properties, including the rewriting cost, optimal ways to change rank-

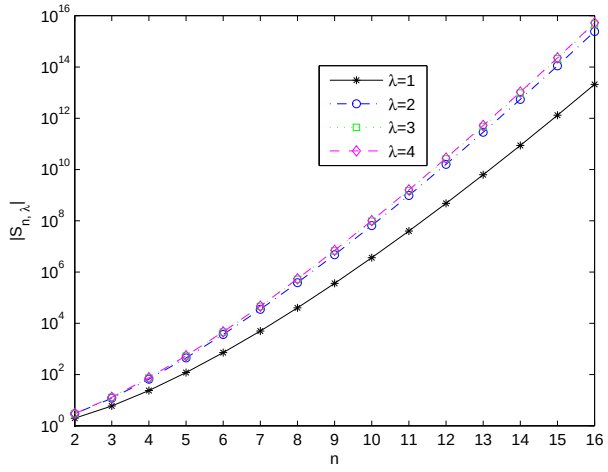

Fig. 1. The value of $|\mathcal{S}_{n,\lambda}|$ for $\lambda = 1, 2, 3, 4$.

modulation states, and the expansion of rank-modulation states given the rewriting cost.

II. Rank Modulation with Multiplicity λ

In this section, we define the concepts on rank modulation with multiplicity λ – in particular those related to rewriting – and study some basic properties.

A. Basic Concepts

The rank modulation with multiplicity λ uses the elements in $\mathcal{S}_{n,\lambda}$, called *rank-modulation states*, to represent data. Let $\mathcal{L} = \{0, 1, \ldots, L-1\}$ denote the alphabet of the stored data. Then there is a surjective map $\mathcal{D} : \mathcal{S}_{n,\lambda} \to \mathcal{L}$, such that the rank-modulation state $\mathbf{s} = (s_1, s_2, \ldots, s_k) \in \mathcal{S}_{n,\lambda}$ represents the data $\mathcal{D}(\mathbf{s}) \in \mathcal{L}$. The number of stored information bits, $\log_2 L$, can be maximized by letting $L = |\mathcal{S}_{n,\lambda}|$; and by letting $L < |\mathcal{S}_{n,\lambda}|$, the cost of rewriting data can be reduced.

Example 1. *Let* $n = 3, \lambda = 2$. *Then* $\mathcal{S}_{n,\lambda} = \{(\{1\},\{2\},\{3\}), (\{1\},\{3\},\{2\}), (\{2\},\{1\},\{3\}), (\{2\},\{3\},\{1\}), (\{3\},\{1\},\{2\}), (\{3\},\{2\},\{1\}), (\{1\},\{2,3\}), (\{2\},\{1,3\}), (\{3\},\{1,2\}), (\{1,2\},\{3\}), (\{1,3\},\{2\}), (\{2,3\},\{1\})\}$. *So* $|\mathcal{S}_{3,2}| = 12$. *Up to* $\log_2 12$ *information bits can be stored.*

The general value of $|\mathcal{S}_{n,\lambda}|$ can be computed by recursion: $|\mathcal{S}_{n,\lambda}| = \sum_{i=1}^{\min\{n,\lambda\}} \binom{n}{i} |\mathcal{S}_{n-i,\lambda}|$ for $n > 0$; and $|\mathcal{S}_{0,\lambda}| = 1$. We show $|\mathcal{S}_{n,\lambda}|$ for $2 \le n \le 16$ and $\lambda = 1, 2, 3, 4$ in Fig. 1.

For the rewriting of data, we consider the memory model where the cell levels can only increase, not decrease [3]. For flash memories, this is the way cells are programmed via charge injection (without the expensive block erasure operation). For PCMs, when the cells are programmed to only become more and more crystallized (without the RESET operation), the same model can be applied. Let us define the basic operation we can use to change the rank-modulation state, in order to rewrite data. The basic operation is a "push operation", where we either push a cell to a higher rank (if there are fewer than λ cells of that rank), or push the cell to the top so that it has a higher rank than all the other $n-1$ cells. More specifically, let $\mathbf{s} = (s_1, s_2, \ldots, s_k) \in \mathcal{S}_{n,\lambda}$ be a rank-modulation state. For any i, j such that $1 \le i < j \le k$ and $|s_i| < \lambda$, if $|s_j| > 1$, with a push operation, we can change $\mathbf{s}$ to

$$(s_1, \ldots, s_i \cup \{p\}, \ldots, s_j \setminus \{p\}, \ldots, s_k)$$

for some $p \in s_j$; if $|s_j| = 1$, we can change $\mathbf{s}$ to

$$(s_1, \ldots, s_i \cup \{p\}, \ldots, s_{j-1}, s_{j+1}, \ldots, s_k)$$

with p being the only element in s_j. And for any $i \in \{1, 2, \ldots, k\}$ such that $|s_i| > 1$, we can change $\mathbf{s}$ to

$$(\{p\}, s_1, \ldots, s_i \setminus \{p\}, \ldots, s_k)$$

for some $p \in s_i$. For any $i \in \{2, 3, \ldots, k\}$ such that $|s_i| = 1$, we can change $\mathbf{s}$ to

$$(\{p\}, s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_k)$$

with p being the only element in s_i. (Note that if $\lambda = 1$, the push operation here is reduced to the "push-to-top" operation for the original rank modulation scheme [3].)

For rewriting data, it is desirable to increase the cell levels as little as possible with each rewrite, so that more rewrites can be performed before the cell levels reach the maximum limit. (After that, the block erasure or RESET operation will be needed to lower the cell levels back to the minimum value.) So in this section, we consider the cost of changing the rank-modulation state from $\mathbf{s}$ to $\mathbf{s}'$ as the minimum number of push operations needed to change $\mathbf{s}$ to $\mathbf{s}'$, which we denote by $d(\mathbf{s}, \mathbf{s}')$. We call $d(\mathbf{s}, \mathbf{s}')$ the *unweighted rewriting cost*. (A weighted version of the rewriting cost will be studied in the next section.) It is not hard to see that

$$\max_{\mathbf{s},\mathbf{s}' \in \mathcal{S}_{n,\lambda}} d(\mathbf{s}, \mathbf{s}') = n - 1.$$

An example of $\mathbf{s}$ and $\mathbf{s}'$ that achieve this maximum unweighted rewriting cost, $d(\mathbf{s}, \mathbf{s}') = n-1$, is $\mathbf{s} = (\{1\}, \ldots, \{i-1\}, \{i\}, \{i+1\}, \ldots, \{n\})$ and $\mathbf{s}' = (\{1\}, \ldots, \{i-1\}, \{i+1\}, \ldots, \{n\}, \{i\})$ for some $1 \le i < n$. (Every cell except c_i needs to be pushed once to change $\mathbf{s}$ to $\mathbf{s}'$.)

B. Unweighted Rewriting Cost

Given two rank-modulation states $\mathbf{s}, \mathbf{s}' \in \mathcal{S}_{n,\lambda}$, we consider how to compute the unweighted rewriting cost $d(\mathbf{s}, \mathbf{s}')$, and how to change $\mathbf{s}$ to $\mathbf{s}'$ with this minimum number of push operations. For the special case $\lambda = 1$, the answer is known [3]: given $\mathbf{s} = (s_1, s_2, \ldots, s_n)$ and $\mathbf{s}' = (s'_1, s'_2, \ldots, s'_n)$, let $\phi : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$ be a bijective map such that for $i = 1, 2, \ldots, n$, we have $s'_i = s_{\phi(i)}$; let r be the minimum integer in $\{1, 2, \ldots, n\}$ such that

$$\phi(r+1) < \phi(r+2) < \cdots < \phi(n);$$

then we have $d(\mathbf{s}, \mathbf{s}') = r$, and the way to change the rank-modulation state from $\mathbf{s}$ to $\mathbf{s}'$ with r push operations is to sequentially pushed the cells with their indices in $s'_r, s'_{r-1}, \ldots, s'_1$ to the top.

For the case $\lambda \geq 2$, we use a tool called *virtual levels*.

Definition 2. *Given a rank-modulation state* $\mathbf{s} = (s_1, s_2, \ldots, s_k) \in \mathcal{S}_{n,\lambda}$, *a "realization" of* $\mathbf{s}$ *is a vector*

$$(v_1, v_2, \ldots, v_n) \in \mathbb{N}^n$$

that satisfies two conditions: (1) $\forall\ 1 \leq i \leq k$ *and* $j_1, j_2 \in s_i$, *we have* $v_{j_1} = v_{j_2}$; *(2)* $\forall\ 1 \leq i_1 < i_2 \leq k$, $j_1 \in s_{i_1}$ *and* $j_2 \in s_{i_2}$, *we have* $v_{j_1} > v_{j_2}$. *We call* v_i *the "virtual level" of the cell* c_i, *for* $i = 1, 2, \ldots, n$.

Definition 3. *Let* $\mathbf{v} = (v_1, v_2, \ldots, v_n)$ *be a realization of* $\mathbf{s} \in \mathcal{S}_{n,\lambda}$, *and let* $\mathbf{v}' = (v'_1, v'_2, \ldots, v'_n)$ *be a realization of* $\mathbf{s}' \in \mathcal{S}_{n,\lambda}$. *The Hamming distance between* $\mathbf{v}$ *and* $\mathbf{v}'$, *denoted by* $H(\mathbf{v}, \mathbf{v}')$, *is*

$$H(\mathbf{v}, \mathbf{v}') = \left|\{i \mid 1 \leq i \leq n,\ v_i \neq v'_i\}\right|.$$

And we say "$\mathbf{v}'$ *dominates* $\mathbf{v}$*" if two conditions are satisfied: (1) for* $i = 1, 2, \ldots, n$, *we have* $v'_i \geq v_i$; *(2) we have*

$$\{v'_i \mid 1 \leq i \leq n, v'_i \leq \max_{1 \leq j \leq n} v_j\} \subseteq \{v_1, v_2, \ldots, v_n\}.$$

We denote "$\mathbf{v}'$ *dominates* $\mathbf{v}$*" by* $\mathbf{v}' \geq \mathbf{v}$.

Lemma 4. *Let* $\lambda \geq 2$. *Let* $\mathbf{s}, \mathbf{s}' \in \mathcal{S}_{n,\lambda}$ *be two rank-modulation states, let* $\mathbf{v} = (v_1, v_2, \ldots, v_n)$ *be a realization of* $\mathbf{s}$, *and let* x *be a non-negative integer. Then,* $\mathbf{s}$ *can be changed into* $\mathbf{s}'$ *by at most* x *push operations if and only if there exists a realization* $\mathbf{v}' = (v'_1, v'_2, \ldots, v'_n)$ *of* $\mathbf{s}'$ *such that* $\mathbf{v}' \geq \mathbf{v}$ *and* $H(\mathbf{v}, \mathbf{v}') \leq x$.

Proof: First, assume that $\mathbf{s}$ can be changed into $\mathbf{s}'$ by $y \leq x$ push operations. We will construct a corresponding realization $\mathbf{v}'$ of $\mathbf{s}'$ as follows. Initially, for $i = 1, 2, \ldots, n$, let $v'_i = v_i$. Then for $i = 1, 2, \ldots, y$, if the i-th push operation pushes a cell c_{j_1} to the same rank as another cell c_{j_2}, then assign to $v_{j'_1}$ the value of $v_{j'_2}$. Otherwise, the i-th push operation pushes a cell c_j to a rank that is higher than all the other $n-1$ cells; in this case, let $z = \max_{1 \leq b \leq n} v'_b$, and we assign to v'_j the value $z+1$. Then, let $\mathbf{v}' = (v'_1, v'_2, \ldots, v'_n)$. It is simple to see that $\mathbf{v}'$ is a realization of $\mathbf{s}'$ and $\mathbf{v}' \geq \mathbf{v}$. Since at most y cells are pushed, at least $n - y$ cells have the same virtual levels in $\mathbf{v}$ and $\mathbf{v}'$; so we have $H(\mathbf{v}, \mathbf{v}') \leq y \leq x$.

Now consider the other direction. Assume that there exists a realization $\mathbf{v}' = (v'_1, v'_2, \ldots, v'_n)$ of $\mathbf{s}'$ such that $\mathbf{v}' \geq \mathbf{v}$ and $H(\mathbf{v}, \mathbf{v}') \leq x$. We will show how to change $\mathbf{s}$ to $\mathbf{s}'$ with $H(\mathbf{v}, \mathbf{v}')$ push operations. We first partition $\{v'_1, v'_2, \ldots, v'_n\}$ into two subsets A and B as follows:

$$A = \{v'_i \mid 1 \leq i \leq n, v'_i > \max_{1 \leq j \leq n} v_j\};$$

$$B = \{v'_i \mid 1 \leq i \leq n, v'_i \leq \max_{1 \leq j \leq n} v_j\}.$$

Since $\mathbf{v}' \geq \mathbf{v}$, we know that $B \subseteq \{v_1, v_2, \ldots, v_n\}$. Here B is the set of virtual levels that are retained when we change $\mathbf{s}$ into $\mathbf{s}'$, and A is the set of virtual levels in $\mathbf{v}'$ that are higher than any virtual level in $\mathbf{v}$. For convenience, we shall denote A as $A = \{a_1, a_2, \ldots, a_{|A|}\}$ such that

$$a_1 < a_2 < \cdots < a_{|A|},$$

and denote B as $B = (b_1, b_2, \ldots, b_{|B|})$ such that

$$b_1 > b_2 > \cdots > b_{|B|}.$$

We change the rank-modulation state from $\mathbf{s}$ to $\mathbf{s}'$ as follows. Initially, for $i = 1, 2, \ldots, n$, let the cell c_i have the virtual level v_i. We will push the cells to higher virtual levels, and the rank-modulation state – which is determined by the virtual levels of the n cells – will change accordingly. We push the cells using the following two steps:

1) For $i = 1, 2, \ldots, |A|$, push the cells in
$$\{c_j \mid 1 \leq j \leq n, v'_j = a_i\}$$
to the virtual level a_i.
2) For $i = 1, 2, \ldots, |B|$, push the cells in
$$\{c_j \mid 1 \leq j \leq n, v_j < v'_j = b_i\}$$
to the virtual level b_i.

During the above two steps, we will use the following method to make sure that for $i = 1, 2, \ldots, |B|$, there is always at least one cell of the virtual level b_i:

- When we are to push a cell c_i from the virtual level $j_1 \in B$ to $j_2 > j_1$, if c_i is the only cell of virtual level j_1 at that moment, then before pushing c_i, we first push a cell in $\{c_z \mid 1 \leq z \leq n, v'_z = j_1\}$ to the virtual level j_1. (Note that if *that* cell is also the only cell of its own virtual level at that moment, then the same rule applies. So there can be a chain reaction of cell pushing of this type. But this chain reaction will stop somewhere because the virtual level of the concerned cell keeps decreasing.)

In the above process, we push every cell at most once.

When the above process ends, the cells have virtual levels $(v'_1, v'_2, \ldots, v'_n)$, which is a realization of $\mathbf{s}'$. A cell c_i $(1 \leq i \leq n)$ is pushed if and only if $v_i \neq v'_i$; and if it is pushed, it is pushed directly to the virtual level v'_i. So the number of push operations equals $H(\mathbf{v}, \mathbf{v}')$. We now show that these $H(\mathbf{v}, \mathbf{v}')$ push operations are all valid operations for the rank-modulation states. Step 1) consists of the "push-to-top" operations, and we sequentially push the cells to higher and higher ranks; clearly, the number of cells at the virtual level a_i (for $1 \leq i \leq |A|$) is never more than λ at any moment. Step 2) consists of the operations that push a cell to a higher and existing rank; and since we process the virtual levels $b_1, b_2, \ldots, b_{|B|}$ sequentially (from high to low), when we process the virtual level b_i (for $1 \leq i \leq |B|$), all the cells that are originally at level b_i have already been pushed up; so as we push cells from below into the level b_i, there will be no more than λ cells in that level. So we have changed $\mathbf{s}$ into $\mathbf{s}'$ with $H(\mathbf{v}, \mathbf{v}') \leq x$ valid push operations. ∎

Theorem 5. *Let* $\lambda \geq 2$. *Let* $\mathbf{s} = (s_1, s_2, \ldots, s_k) \in \mathcal{S}_{n,\lambda}$ *and* $\mathbf{s}' = (s'_1, s'_2, \ldots, s'_{k'}) \in \mathcal{S}_{n,\lambda}$ *be two rank-modulation states,*

let $\mathbf{v} = (v_1, v_2, \ldots, v_n)$ *be a realization of* $\mathbf{s}$*, and define* V *as* $V = \{\mathbf{u} \mid \mathbf{u}$ *is a realization of* $\mathbf{s}'$*,* $\mathbf{u} \geq \mathbf{v}\}$*. Then we have*

$$d(\mathbf{s}, \mathbf{s}') = \min_{\mathbf{u} \in V} H(\mathbf{v}, \mathbf{u}).$$

Furthermore, define $\mathbf{v}' = (v'_1, v'_2, \ldots, v'_n)$ *as follows:*

1) *Let* $h_{k'} = \max_{j \in s'_{k'}} v_j$.
   $\forall\, i \in s'_{k'}$*, let* $v'_i = h_{k'}$.
2) *For* $i_1 = k' - 1, k' - 2, \ldots, 1$*, do:*
   - *If* $\max_{j \in s'_{i_1}} v_j > h_{i_1+1}$*, then let* $h_{i_1} = \max_{j \in s'_{i_1}} v_j$*; if* $\max_{j \in s'_{i_1}} v_j \leq h_{i_1+1} < \max_{1 \leq j \leq n} v_j$*, then let* $h_{i_1} = \min\{v_j \mid 1 \leq j \leq n, v_j > h_{i_1+1}\}$*; if* $\max_{j \in s'_{i_1}} v_j \leq h_{i_1+1}$ *and* $h_{i_1+1} \geq \max_{1 \leq j \leq n} v_j$*, then let* $h_{i_1} = h_{i_1+1} + 1$.
   - $\forall\, i_2 \in s'_{i_1}$*, let* $v'_{i_2} = h_{i_1}$.

Then we have $\mathbf{v}' \in V$ *and* $H(\mathbf{v}, \mathbf{v}') = \min_{\mathbf{u} \in V} H(\mathbf{v}, \mathbf{u})$.

Proof: Lemma 4 leads to $d(\mathbf{s}, \mathbf{s}') = \min_{\mathbf{u} \in V} H(\mathbf{v}, \mathbf{u})$. When we assign values to $(v'_1, v'_2, \ldots, v'_n)$ (which are virtual levels for the n cells corresponding to the rank-modulation state $\mathbf{s}'$), we are sequentially assigning virtual levels to the cells with indices in $s'_{k'}, s'_{k'-1}, \ldots, s'_1$; and for $i = k', k' - 1, \ldots, 1$, we give the cells with indices in s'_i a virtual level that is as small as possible, as long as the condition $\mathbf{v}' \in V$ is satisfied. A proof by induction can show that compared to all the realizations of $\mathbf{s}'$ in V, here each h_i $(1 \leq i \leq k')$ – and therefore each virtual level v'_i $(1 \leq i \leq n)$ – is individually minimized, and a cell is pushed only when necessary. (Since the cells are pushed only upward, minimizing h_i is a greedy and optimal approach for minimizing $h_{i-1}, h_{i-2}, \ldots, h_1$ and for minimizing the number of cells that need to be pushed.) So $H(\mathbf{v}, \mathbf{v}') = \min_{\mathbf{u} \in V} H(\mathbf{v}, \mathbf{u})$. ■

Theorem 5 shows how to find the realization $\mathbf{v}'$ for $\mathbf{s}'$ such that $\mathbf{v}'$ dominates $\mathbf{v}$ (the realization of $\mathbf{s}$) and $H(\mathbf{v}, \mathbf{v}') = d(\mathbf{s}, \mathbf{s}')$. The proof of Lemma 4 shows given such a realization $\mathbf{v}$, how to change the rank-modulation state from $\mathbf{s}$ to $\mathbf{s}'$ with $d(\mathbf{s}, \mathbf{s}')$ push operations. By combining them, we can not only compute $d(\mathbf{s}, \mathbf{s}')$, but also transform $\mathbf{s}$ to $\mathbf{s}'$ with the minimum unweighted rewriting cost. For simplicity, we skip the presentation of the algorithm. We show an example below.

Example 6. *Suppose* $\lambda = 2$*,* $n = 8$*,* $\mathbf{s} = (\{2,3\}, \{7\}, \{4,5\}, \{1,8\}, \{6\})$*,* $\mathbf{s}' = (\{2,3\}, \{4\}, \{1\}, \{7,8\}, \{5,6\})$*. We let* $\mathbf{v} = (2,5,5,3,3,1,4,2)$ *be a realization of* $\mathbf{s}$*. (See Fig.* 2*.) Then by Theorem* 5*, we get the realization* $\mathbf{v}' = (5,7,7,6,3,3,4,4)$ *of* $\mathbf{s}'$*. (It can be seen that* $\mathbf{v}' \geq \mathbf{v}$*.) So we get* $d(\mathbf{s}, \mathbf{s}') = H(\mathbf{v}, \mathbf{v}') = 6$*. Then by the steps specified in the proof of Lemma* 4*, we get the 6 push operations that change* $\mathbf{s}$ *into* $\mathbf{s}'$*. (See Fig.* 2*, where the push operations are shown as arrows, and the numbers beside arrows represent their order.)*

C. Sizes of Spheres

For a rank-modulation state $\mathbf{s} \in \mathcal{S}_{n,\lambda}$ and an unweighted rewriting cost $r \geq 0$, we define the *sphere of unweighted radius* r *centered at* $\mathbf{s}$ as $\theta(\mathbf{s}, r) \triangleq \{\mathbf{u} \in \mathcal{S}_{n,\lambda} \mid d(\mathbf{s}, \mathbf{u}) = r\}$, and define the *ball of unweighted radius* r *centered at* $\mathbf{s}$ as $\beta(\mathbf{s}, r) \triangleq \{\mathbf{u} \in \mathcal{S}_{n,\lambda} \mid d(\mathbf{s}, \mathbf{u}) \leq r\}$. Clearly, $|\beta(\mathbf{s}, r)| = \sum_{i=0}^{r} |\theta(\mathbf{s}, i)|$. Knowing the sizes of spheres and balls is useful for analyzing the performance of rewriting. For example, when the states in $\mathcal{S}_{n,\lambda}$ are used to represent data of the alphabet $\mathcal{L}$, if the rank-modulation state is currently $\mathbf{s} \in \mathcal{S}_{n,\lambda}$, for the next rewrite, the unweighted rewriting cost in the worst case is at least $\min\{r \mid r \geq 0, |\beta(\mathbf{s}, r)| \geq |\mathcal{L}|\}$.

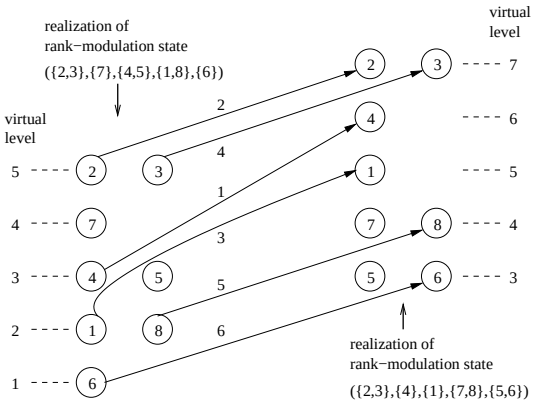

Fig. 2. Change rank-modulation state from $\mathbf{s}$ to $\mathbf{s}'$ with $d(\mathbf{s}, \mathbf{s}')$ pushes.

We show how to compute $|\theta(\mathbf{s}, r)|$ for $\mathbf{s} \in \mathcal{S}_{n,\lambda}$ and $0 \leq r \leq n-1$. If $\lambda = 1$, we have $|\theta(\mathbf{s}, r)| = \frac{n!}{(n-r)!} - \frac{n!}{(n-r+1)!}$ for $1 \leq r \leq n-1$ and $|\theta(\mathbf{s}, 0)| = 1$ [3]. So in the following, we consider $\lambda \geq 2$. Fix a realization $\mathbf{v} = (v_1, v_2, \ldots, v_n)$ for $\mathbf{s} = (s_1, s_2, \ldots, s_k)$ – say the realization where the n cells have virtual levels from 1 to k – and we see that for any $\mathbf{s}' \in \mathcal{S}_{n,\lambda}$, Theorem 5 finds a unique realization $\mathbf{v}' = (v'_1, v'_2, \ldots, v'_n)$ for $\mathbf{s}'$ such that $\mathbf{v}' \geq \mathbf{v}$, $H(\mathbf{v}, \mathbf{v}') = d(\mathbf{s}, \mathbf{s}')$ and every virtual level v'_i $(1 \leq i \leq n)$ is minimized. So to compute $|\theta(\mathbf{s}, r)|$, the number of states in the sphere $\theta(\mathbf{s}, r)$, we can equivalently compute the number of such unique realizations (of the states in $\theta(\mathbf{s}, r)$), because they have a one to one correspondence.

Let $\sigma_1, \sigma_2, \ldots, \sigma_\kappa$ and X be $\kappa + 1$ mutually disjoint sets of cells, where $1 \leq |\sigma_i| \leq \lambda$ for $1 \leq i \leq \kappa$ and $|X| = x \in \{0, 1, \ldots, n-1\}$. For $i = 1, \ldots, \kappa$, we assign the virtual level $\kappa + 1 - i$ to the cells in the set σ_i. Let $\delta \in \{0, 1, \ldots, n-1\}$, $t \in \{1, 2, \ldots, \lambda\}$, $\gamma \in \{x, x+1, \ldots, n-1\}$ and $tag \in \{0, 1\}$ be given parameters. Let $\mathcal{R}$ denote the set of realizations (that is, assignments of virtual levels to the $x + \sum_{i=1}^{\kappa} |\sigma_i|$ cells) that we can change this current realization into, given the following constraints:

1) We obtain a realization in $\mathcal{R}$ by pushing $\gamma - x$ cells in $\cup_{i=1}^{\kappa} \sigma_i$ to higher virtual levels, and by assigning the x cells in X to the virtual levels between 1 and $\kappa + \delta$. Every cell is pushed or assigned at most once. For the realization in $\mathcal{R}$, every virtual level has at most λ cells.
2) For a realization in $\mathcal{R}$, the maximum virtual level that has a cell is level $\kappa + \delta$, and exactly t cells are in that virtual level $\kappa + \delta$.
3) For a realization in $\mathcal{R}$, if a cell in $\cup_{i=1}^{\kappa} \sigma_i$ is pushed to a level $j \in \{2, 3, \ldots, \kappa + \delta\}$, or if a cell in X is assigned to a level $j \in \{2, 3, \ldots, \kappa + \delta\}$, then for this realization

in $\mathcal{R}$, either some cell is in the virtual level $j-1$, or $2 \le j \le \kappa$ and some cell in $\sigma_{\kappa+1-j}$ is in the level j.

4) If $tag = 1$, then no cell in X can be assigned to the virtual level 1 unless for this realization in $\mathcal{R}$, some cell in σ_κ is in the virtual level 1.

We use $f(|\sigma_1|, |\sigma_2|, \ldots, |\sigma_\kappa|; x; \delta; t; \gamma; tag)$ to denote the cardinality of $\mathcal{R}$. We can see that the sphere size

$$|\theta(\mathbf{s}, r)| = \sum_{\delta=0}^{r} \sum_{t=1}^{\lambda} f(|s_1|, |s_2|, \ldots, |s_k|; 0; \delta; t; r; 0).$$

We show how to use recursion to compute the value of $f(|\sigma_1|, |\sigma_2|, \ldots, |\sigma_\kappa|; x; \delta; t; \gamma; tag)$. For simplicity, we only introduce the main recursion, and skip introducing the values of $f(|\sigma_1|, |\sigma_2|, \ldots, |\sigma_\kappa|; x; \delta; t; \gamma; tag)$ for the boundary cases. (The boundary values can be obtained easily.)

To change the given realization to a realization in $\mathcal{R}$, say that we push y_1 cells in σ_κ to the maximum virtual level $k+\delta$, push y_2 cells in σ_κ to the virtual levels $2, 3, \ldots, k+\delta-1$, and assign y_3 cells in X to the virtual level 1. Note that once y_1, y_2, y_3 are fixed, the number of cells in level 1 becomes fixed, and we do not need to consider it furthermore. So we get the recursion:

- If $tag = 0$, then let $P_1 \triangleq \{(y_1, y_2, y_3) \in \mathbb{Z}^3 \mid 0 \le y_1 < t; 0 \le y_2 \le |\sigma_\kappa|; 0 \le y_3 \le \min\{x, \lambda - |\sigma_\kappa| + y_1 + y_2\}$; either "$y_1 + y_2 < |\sigma_\kappa|$" or "$y_1 + y_2 = |\sigma_\kappa|$ and $y_3 > 0$"$\}$, let $P_2 \triangleq \{(y_1, y_2, y_3) \in \mathbb{Z}^3 \mid 0 \le y_1 \le \min\{t-1, |\sigma_\kappa|\}; y_2 = |\sigma_\kappa| - y_1; y_3 = 0\}$, let $P_3 \triangleq \{(y_1, y_2, y_3) \in \mathbb{Z}^3 \mid y_1 = t; 0 \le y_2 \le |\sigma_\kappa|; 0 \le y_3 \le \min\{x, \lambda - |\sigma_\kappa| + y_1 + y_2\}$; either "$y_1 + y_2 < |\sigma_\kappa|$" or "$y_1 + y_2 = |\sigma_\kappa|$ and $y_3 > 0$"$\}$, and let $P_4 \triangleq \{(y_1, y_2, y_3) \in \mathbb{Z}^3 \mid y_1 = t; y_2 = |\sigma_\kappa| - t \ge 0; y_3 = 0\}$.
  If $tag = 1$, then let $P_1 \triangleq \{(y_1, y_2, y_3) \in \mathbb{Z}^3 \mid 0 \le y_1 < t; 0 \le y_2 < |\sigma_\kappa| - y_1; 0 \le y_3 \le \min\{x, \lambda - |\sigma_\kappa| + y_1 + y_2\}\}$, let $P_2 \triangleq \{(y_1, y_2, y_3) \in \mathbb{Z}^3 \mid 0 \le y_1 \le \min\{t-1, |\sigma_\kappa|\}; y_2 = |\sigma_\kappa| - y_1; y_3 = 0\}$, let $P_3 \triangleq \{(y_1, y_2, y_3) \in \mathbb{Z}^3 \mid y_1 = t; 0 \le y_2 < |\sigma_\kappa| - t; 0 \le y_3 \le \min\{x, \lambda - |\sigma_\kappa| + y_1 + y_2\}\}$, and let $P_4 \triangleq \{(y_1, y_2, y_3) \in \mathbb{Z}^3 \mid y_1 = t; y_2 = |\sigma_\kappa| - t \ge 0; y_3 = 0\}$.
- We have $f(|\sigma_1|, |\sigma_2|, \ldots, |\sigma_\kappa|; x; \delta; t; \gamma; tag) = \sum_{(y_1,y_2,y_3)\in P_1} \binom{|\sigma_\kappa|}{y_1}\binom{|\sigma_\kappa|-y_1}{y_2}\binom{x}{y_3} f(|\sigma_1|, |\sigma_2|, \cdots, |\sigma_{\kappa-1}|; x + y_2 - y_3; \delta; t - y_1; \gamma - y_1 - y_3; 0) + \sum_{(y_1,y_2,y_3)\in P_2} \binom{|\sigma_\kappa|}{y_1}\binom{|\sigma_\kappa|-y_1}{y_2}\binom{x}{y_3} f(|\sigma_1|, |\sigma_2|, \cdots, |\sigma_{\kappa-1}|; x + y_2; \delta; t - y_1; \gamma - y_1; 1) + \sum_{(y_1,y_2,y_3)\in P_3} \binom{|\sigma_\kappa|}{y_1}\binom{|\sigma_\kappa|-y_1}{y_2}\binom{x}{y_3} \sum_{1\le z\le\lambda} f(|\sigma_1|, |\sigma_2|, \cdots, |\sigma_{\kappa-1}|; x + y_2 - y_3; \delta - 1; z; \gamma - y_1 - y_3; 0) + \sum_{(y_1,y_2,y_3)\in P_4} \binom{|\sigma_\kappa|}{y_1}\binom{|\sigma_\kappa|-y_1}{y_2}\binom{x}{y_3} \sum_{1\le z\le\lambda} f(|\sigma_1|, |\sigma_2|, \cdots, |\sigma_{\kappa-1}|; x + y_2; \delta - 1; z; \gamma - y_1; 1)$.

Given any $\mathbf{s} \in \mathcal{S}_{n,\lambda}$ and $r \le n-1$, the time complexity of computing the sphere size $|\theta(\mathbf{s}, r)|$ using the above recursion is $O(n^4\lambda^5)$. Due to the space limitation, we skip the proof of the following theorem.

Theorem 7. *The above recursion correctly computes* $|\theta(\mathbf{s}, r)|$.

III. Weighted Rewriting Cost

We have studied the unweighted rewriting cost, where every push operation is considered to have cost one. In practice, however, the operations can have different cost values: a push operation that increases the cell level less is more preferable than a push operation that increases the cell level more. So in this section, we present the definition of *weighted rewriting cost*, which measures the cost of push operations based on how much they increase the cell levels.

As a combinatorial definition, we use the help of virtual levels. Let $\mathbf{s} = (s_1, s_2, \ldots, s_k) \in \mathcal{S}_{n,\lambda}$ and $\mathbf{s}' \in \mathcal{S}_{n,\lambda}$ be two rank-modulation states. Let $\mathbf{v} = (v_1, v_2, \ldots, v_n)$ be the unique realization of $\mathbf{s}$ such that $\{v_1, v_2, \ldots, v_n\} = \{1, 2, \ldots, k\}$. Let $V \triangleq \{\mathbf{u} \mid \mathbf{u}$ is a realization of $\mathbf{s}', \mathbf{u} \ge \mathbf{v}\}$. By the previous analysis, we know that a sequence of push operations that changes the rank-modulation state from $\mathbf{s}$ to $\mathbf{s}'$ also changes the realization from $\mathbf{v}$ to some $\mathbf{u} \in V$ (and vice versa). Virtual levels are a reasonable simplification of real cell levels. So we define the weighted rewriting cost of changing $\mathbf{s}$ into $\mathbf{s}'$ as

$$w(\mathbf{s}, \mathbf{s}') = \min_{(u_1,u_2,\ldots,u_n)\in V} \sum_{i=1}^{n} (u_i - v_i).$$

Let $\mathbf{v}' = (v'_1, v'_2, \ldots, v'_n)$ be the unique realization of $\mathbf{s}'$ that is generated by Theorem 5. It has been shown that $\mathbf{v}'$ minimizes the virtual level of every cell; so we have

$$w(\mathbf{s}, \mathbf{s}') = \sum_{i=1}^{n} (v'_i - v_i) = \sum_{i=1}^{n} \min_{(u_1,\ldots,u_n)\in V} (u_i - v_i).$$

And it is not hard to see that $\max_{\mathbf{s},\mathbf{s}'\in\mathcal{S}_{n,\lambda}} w(\mathbf{s}, \mathbf{s}') = n(n-1)$.

Given a state $\mathbf{s} \in \mathcal{S}_{n,\lambda}$ and an integer $r \ge 0$, we can define the *sphere of weighted radius r centered at* $\mathbf{s}$ as $\Theta(\mathbf{s}, r) \triangleq \{\mathbf{u} \in \mathcal{S}_{n,\lambda} \mid w(\mathbf{s}, \mathbf{u}) = r\}$. The sphere size, $|\Theta(\mathbf{s}, r)|$, can be computed with a similar recursion as the one in the previous section. For simplicity we skip the details.

Acknowledgment

This work was supported in part by the NSF CAREER Award CCF-0747415 and the NSF grant ECCS-0802107.

References

[1] A. Barg and A. Mazumdar, "Codes in permutations and error correction for rank modulation," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, pp. 854-858, June 2010.

[2] F. Zhang, H. Pfister and A. Jiang, "LDPC codes for rank modulation in flash memories," in *Proc. ISIT*, pp. 859-863, June 2010.

[3] A. Jiang, R. Mateescu, M. Schwartz and J. Bruck, "Rank modulation for flash memories," in *IEEE Transactions on Information Theory*, vol. 55, no. 6, pp. 2659-2673, June 2009.

[4] A. Jiang, M. Schwartz and J. Bruck, "Correcting charge-constrained errors in the rank modulation scheme," in *IEEE Transactions on Information Theory*, vol. 56, no. 5, pp. 2112-2120, May 2010.

[5] M. Schwartz, "Constant-weight Gray codes for local rank modulation," in *Proc. ISIT*, pp. 869-873, June 2010.

[6] I. Tamo and M. Schwartz, "Correcting limited-magnitude errors in the rank-modulation scheme," in *Proc. ITA Workshop*, Feb. 2010.

[7] Z. Wang and J. Bruck, "Partial rank modulation for flash memories," in *Proc. ISIT*, pp. 864-868, June 2010.

[8] Z. Wang, A. Jiang and J. Bruck, "On the capacity of bounded rank modulation for flash memories," in *Proc. ISIT*, pp. 1234-1238, 2009.