Error-Correcting Codes for Rank Modulation

Anxiao (Andrew) Jiang
Computer Science Department
Texas A&M University
College Station, TX 77843, U.S.A.
ajiang@cs.tamu.edu

Moshe Schwartz
Electrical and Computer Engineering
Ben-Gurion University
Beer Sheva 84105, Israel
schwartz@ee.bgu.ac.il

Jehoshua Bruck
Electrical Engineering Department
California Institute of Technology
Pasadena, CA 91125, U.S.A.
bruck@paradise.caltech.edu

***Abstract*—We investigate error-correcting codes for a novel storage technology for flash memories, the rank-modulation scheme. In this scheme, a set of n cells stores information in the permutation induced by the different charge levels of the individual cells. The resulting scheme eliminates the need for discrete cell levels, overcomes overshoot errors when programming cells (a serious problem that reduces the writing speed), and mitigates the problem of asymmetric errors.**

In this paper, we study the properties of error correction in rank modulation codes. We show that the adjacency graph of permutations is a subgraph of a multi-dimensional array of a special size, a property that enables code designs based on Lee-metric codes. We present a one-error-correcting code whose size is at least half of the optimal size. We also present additional error-correcting codes and some related bounds.

I. Introduction

Flash memory is an electronic non-volatile memory (NVM) that uses floating-gate cells to store information [2]. In the standard technology, every flash cell has q discrete states – state 0, 1, $\cdots$, $q-1$ – and therefore can store $\log_2 q$ bits. The flash memory changes the state of a cell by injecting or removing charge into/from the cell. To increase a cell from a lower state to a higher state, charge (e.g., electrons for nFETs) is injected into the cell and is trapped there. This operation is called *cell programming*. To decrease a cell's state, charge is removed from the cell, which is called *cell erasing*. Flash memory is widely used in mobile, embedded, and mass-storage systems because of its physical robustness, high density, and good performance [2]. To expand its storage capacity, research on multi-level cells with large values of q is actively underway.

For flash memories, writing is more time- and energy-consuming than reading [2]. The main factor is the iterative cell-programming procedure designed to avoid over-programming [1] (raising the cell's charge level above its target level). In flash memories, cells are organized into blocks, where each block has a large number ($\approx 10^5$) of cells [2]. Cells can be programmed individually, but to decrease the state of a cell, the whole block has to be erased to the lowest state and then re-programmed. Since over-programming can only be corrected by the block erasure, in practice a conservative procedure is used for programming a cell, where charge is injected into the cell over quite a few rounds [1]. After every round, the charge level of the cell is measured and the next-round injection is configured. The charge level of the cell is made to gradually approach the target state until it achieves the desired accuracy. The iterative-programming approach is costly in time and energy.

A second challenge for flash memory is data reliability. The stored data can be lost due to charge leakage, a long-term factor that causes the data retention problem. The data can also be affected by other mechanisms, including read disturbance, write disturbance [2], etc. Many of the error mechanisms have an asymmetric property: they make the numerous cells' charge levels drift in one direction. (For example, charge leakage makes the cell levels drift down.) Such a drift of cell charge levels causes errors in aging devices.

In this paper, we propose and study a new scheme for storing data in flash memories, the *rank-modulation scheme*. It aims at eliminating the risk of cell over-programming, and reducing the effect of asymmetric errors. Given a set of n cells with distinct charge levels, the *rank* of a cell indicates the relative position of its own charge level, and the ranks of the n cells induces a permutation of $\{1, 2, \ldots, n\}$. The rank modulation scheme uses this permutation to store information. To write data into the n cells, we first program the cell with the lowest rank, then the cell with the second lowest rank, and finally the cell with the highest rank. While programming the cell with rank i ($1 < i \leqslant n$), the only requirement is to make its charge level be above that of the cell with rank $i-1$.

The rank-modulation scheme eliminates the need to use the absolute values of cell levels to store information. Instead, the relative ranks are used. Since there is no risk of over-programming and the cell charge levels can take continuous values, a substantially less conservative cell programming method can be used and the writing speed can be improved. In addition, asymmetric errors become less serious, because when cell levels drift in the same direction, their ranks are not affected as much as their absolute values. This way both the writing speed and the data reliability can be improved.

In this paper, we study error-correcting codes for rank modulation. Even though asymmetric drifts of cell levels are tolerated better by rank modulation, errors can still happen because the cell levels do not necessarily drift at the same rate. A companion paper [6] studies Gray codes and encoding/decoding algorithms for the rank modulation scheme.

We explore the properties associated with error-correcting rank-modulation codes. We show that the adjacency graph of permutations for n cells, which is induced by the error model, is a subgraph of a $2 \times 3 \times \cdots \times n$ linear array. This observation establishes a general method for designing error-correcting rank-modulation codes using Lee-metric error-correcting codes. We present a single-error-correcting code whose size is at least half of the maximum size. We also present results on additional error-correcting codes and some related bounds.

The rest of the paper is organized as follows. In Section II some notations are defined. We continue in Section III, to investigate properties associated with permutations and error correction. In Section IV some code constructions are presented, and in Section V, more results on codes are presented. In Section VI, the paper is concluded.

II. Definitions and Notation

Let n flash memory cells be denoted by $1, 2, \dots, n$. For $1 \leqslant i \leqslant n$, let $c_i \in \mathbb{R}$ denote the charge level of cell i. The ranks of the n cells is a permutation of $\{1, 2, \dots, n\}$. If the permutation is $[a_1, a_2, \dots, a_n]$, then $c_{a_1} > c_{a_2} > \cdots > c_{a_n}$. Here the cell a_1 has the highest rank and the cell a_n has the lowest rank.

A rank-modulation scheme uses the ranks (i.e, the permutation) to store information. Let S_n denote the set of $n!$ permutations. Let $Q = \{1, 2, \dots, q\}$ denote the alphabet of the symbol stored in the n cells. The rank-modulation scheme defines a decoding function, $D : S_n \to Q$ which maps permutations (induced by the relative charge levels of the cells) to symbols from the user alphabet.

Given a permutation, an *adjacent transposition* is the local exchange of two adjacent elements in the permutation: $[a_1, \dots, a_{i-1}, a_i, a_{i+1}, a_{i+2}, \dots, a_n]$ is changed to $[a_1, \dots, a_{i-1}, a_{i+1}, a_i, a_{i+2}, \dots, a_n]$.

In this model of representation, the minimal change to a permutation caused by charge-level drift is a single adjacent transposition. We measure the number of errors by the minimum number of adjacent transpositions needed to change the permutation from its original value to its erroneous value. For example, if the errors change the permutation from $[2, 1, 3, 4]$ to $[2, 3, 4, 1]$, the number of errors is two, because at least two adjacent transpositions are needed to change one into the other: $[2, 1, 3, 4] \to [2, 3, 1, 4] \to [2, 3, 4, 1]$.

For two permutations A and B, define their distance, $d(A, B)$, as the minimal number of adjacent transpositions needed to change A into B. This distance measure is called the Kendall Tau Distance in the statistics and machine-learning community [7], and it induces a metric over S_n. If $d(A, B) = 1$, A and B are called *adjacent*. Any two permutations of S_n are at distance at most $\frac{n(n-1)}{2}$ from each other. Two permutations of maximum distance are a reverse of each other.

III. Properties and Bounds

In this section, we study the distance between permutations and the coordinate representation of permutations. We then study the sizes of balls, and derive an upper bound on the cardinality of error-correcting rank-modulation codes.

Theorem 1. *Let $A = [a_1, a_2, \dots, a_n]$ and $B = [b_1, b_2, \dots, b_n]$ be two permutations of length n. Suppose that $b_p = a_n$ for some $1 \leqslant p \leqslant n$. Let $A' = [a_1, a_2, \dots, a_{n-1}]$ and $B' = [b_1, \dots, b_{p-1}, b_{p+1}, \dots, b_n]$. Then,*

$$d(A, B) = d(A', B') + n - p.$$

Proof: Let T be a sequence of $d(A, B)$ adjacent transpositions that change A into B. Divide T into two subsequences T_1 and T_2, such that T_1 contains those adjacent transpositions that involve a_n, and T_2 contains those adjacent transpositions that do not involve a_n. (For instance, let us use $t(a_i, a_j)$ to denote an adjacent transposition that exchanges the two numbers a_i and a_j. Suppose, for example, $A = [2, 3, 1, 4]$, $B = [3, 4, 1, 2]$, and the minimum number of adjacent transpositions change A into B as $[2, 3, 1, 4] \to [3, 2, 1, 4] \to [3, 2, 4, 1] \to [3, 4, 2, 1] \to [3, 4, 1, 2]$. Then T is $t(2, 3)$, $t(1, 4)$, $t(2, 4)$, $t(2, 1)$, T_1 is $t(2, 3)$, $t(2, 1)$, and T_2 is $t(1, 4)$, $t(2, 4)$.) Let $|T|$, $|T_1|$ and $|T_2|$ denote the number of adjacent transpositions in T, T_1 and T_2, respectively. Clearly, $|T| = |T_1| + |T_2|$.

It is not hard to see that T_2 can also change A' into B'. That is because for any $a_i \neq a_n$ and $a_j \neq a_n$, an adjacent transposition in T_1, which involves a_n, does not change the relative positions of a_i and a_j in A' (and its changed version). Meanwhile, an adjacent transposition $t(a_i, a_j)$ in T_2 changes the relative positions of a_i and a_j the same way for A and A' (and their changed versions). Therefore, $|T_2| \geqslant d(A', B')$. It can also be seen that $|T_1| \geqslant n - p$, because every adjacent transposition moves a_n forward in the permutation by one position, and from A to B a_n has moved $n - p$ positions. So $d(A, B) = |T| = |T_1| + |T_2| \geqslant d(A', B') + n - p$.

Now we show that $d(A, B) \leqslant d(A', B') + n - p$. Consider such a sequence of $d(A', B') + n - p$ adjacent transpositions: the first $d(A', B')$ of them change $A = [A', a_n]$ into $[B', a_n]$, and the next $n - p$ of them keep moving a_n forward and thus change $[B', a_n]$ into B. So $d(A, B) \leqslant d(A', B') + n - p$. It follows that $d(A, B) = d(A', B') + n - p$. ■

The above theorem shows a recursive algorithm for computing the distance between two permutations. Let $A = [a_1, a_2, \dots, a_n]$ and $B = [b_1, b_2, \dots, b_n]$ be two permutations. For $1 \leqslant i \leqslant n$, let A_i denote $[a_1, a_2, \dots, a_i]$, let B_i denote the subsequence of B that contains only those numbers in A_i, and let p_i denote the position of a_i in B_i. Then, since $d(A_1, B_1) = 0$ and $d(A_i, B_i) = d(A_{i-1}, B_{i-1}) + i - p_i$, for $i = 2, 3, \dots, n$, we get

$$d(A, B) = d(A_n, B_n) = \frac{(n-1)(n+2)}{2} - \sum_{i=2}^{n} p_i.$$

We now define a coordinate system for permutations. We fix $A = [1, 2, \dots, n]$. For every permutation $B = [b_1, b_2, \dots, b_n]$, we define its *coordinates* as $X_B = (2 - p_2, 3 - p_3, \dots, n - p_n)$. Here p_i is defined as above for $2 \leqslant i \leqslant n$. Clearly, if $X_B = (x_1, x_2, \dots, x_{n-1})$, then $0 \leqslant x_i \leqslant i$ for $1 \leqslant i \leqslant n - 1$.

Example 2. *Let $A = [1, 2, 3, 4, 5]$. Then $X_A = (0, 0, 0, 0)$. If $B = [3, 4, 2, 1, 5]$, then $X_B = (1, 2, 2, 0)$. If $B = [5, 4, 3, 2, 1]$, then $X_B = (1, 2, 3, 4)$. The full set of coordinates for $n = 3$ and $n = 4$ are shown in Fig. 1 (a) and (c), respectively.* □

The coordinate system is equivalent to a form of Lehmer code (or Lucas-Lehmer code, inversion table) [9]. It is easy to see that two permutations are identical if and only if they have the same coordinates, and any vector $(y_1, y_2, \dots, y_{n-1})$, $0 \leqslant y_i \leqslant i$ for $1 \leqslant i \leqslant n - 1$, is the coordinates of some permutation in S_n. So there is a one-to-one correspondence between the coordinates and the permutations.

Let $A \in S_n$ be a permutation. For any $0 \leqslant r \leqslant \frac{n(n-1)}{2}$, the set $\mathcal{B}_r(A) = \{B \in S_n \mid d(A,B) \leqslant r\}$ is a *ball* of radius r centered at A. A simple relabeling argument suffices to show that the size of a ball does not depend on the choice of center. We use $|\mathcal{B}_r|$ to denote $|\mathcal{B}_r(A)|$ for any $A \in S$. We are interested in finding the value of $|\mathcal{B}_r|$. The following theorem presents a way to compute the size of a ball using polynomial multiplication.

Theorem 3. *For $0 \leqslant r \leqslant \frac{n(n-1)}{2}$, let e_r denote the coefficient of x^r in the polynomial $\prod_{i=1}^{n-1} \frac{x^{i+1}-1}{x-1}$. Then $|\mathcal{B}_r| = \sum_{i=0}^{r} e_r$.*

Proof: Let $A = [1,2,\ldots,n]$. Let $B = [b_1,b_2,\ldots,b_n]$ be a generic permutation. Let $X_B = (y_1,y_2,\ldots,y_{n-1})$ be the coordinates of B. By the definition of coordinates, we get $d(A,B) = \sum_{i=1}^{n-1} y_i$. The number of permutations at distance r from A equals the number of integer solutions to $\sum_{i=1}^{n-1} y_i = r$ such that $0 \leqslant y_i \leqslant i$. That is equal to the coefficient of x^r in the polynomial $\prod_{i=1}^{n-1}(x^i + x^{i-1} + \cdots + 1) = \prod_{i=1}^{n-1} \frac{x^{i+1}-1}{x-1}$. Thus, there are exactly e_r permutations at distance r from A, and $|\mathcal{B}_r| = \sum_{i=0}^{r} e_i$. ■

Polynomial multiplication is a well-studied area, and efficient algorithms exist. Theorem 3 induces an upper bound for the sizes of error-correcting rank-modulation codes. By the sphere-packing principle, for such a code that can correct r errors, its size cannot exceed $n!/|\mathcal{B}_r|$.

IV. Error-correcting Rank-Modulation Codes

In this section, we first study the topology of permutations, and use the result to derive a general construction for error-correcting rank-modulation codes based on Lee-metric codes. Next, we present a family of one-error-correcting codes whose size is at least half of the optimal size.

A. Embedding of Permutation Adjacency Graph

Define the adjacency graph of permutations, $G = (V,E)$, as follows. The graph G has $|V| = n!$ vertices, which represent the $n!$ permutations. Two vertices $u, v \in V$ are adjacent if and only if $d(u,v) = 1$. G is a regular undirected graph with degree $n-1$ and diameter $\frac{n(n-1)}{2}$. To study the topology of G, we begin with the follow theorem.

Theorem 4. *For two permutations $A = [a_1,a_2,\ldots,a_n]$ and $B = [b_1,b_2,\ldots,b_n]$, let their coordinates be $X_A = (x_1,x_2,\ldots,x_{n-1})$ and $X_B = (y_1,y_2,\ldots,y_{n-1})$. A and B are adjacent if and only if they satisfy the following two conditions:*

- *Condition 1: $\sum_{i=1}^{n-1} |x_i - y_i| = 1$.*
- *Condition 2: There do not exist $i,j \in \{1,2,\ldots,n\}$, where $i < j-1$, such that (1) $a_i = b_j, a_j = b_i$; (2) for any k where $k \neq i$ and $k \neq j$, $a_k = b_k$; (3) for any k where $i < k < j$, $a_k > b_i$ and $a_k > b_j$.*

Proof: The proof is by induction. When $n = 2$, the theorem is easily true. That serves as the base case.

Now assume that the theorem is true for $n = 2,3,\ldots,N-1$. We will prove that it is also true when $n = N$. First, we will show that if the two permutations are adjacent, then $\sum_{i=1}^{N-1} |x_i - y_i| = 1$.

Suppose A and B are adjacent. Consider the two integers z_1, z_2 such that the z_1-th element in A and the z_2-th element in B are both N. There are two cases. Case 1: $z_1 = z_2$. In this case, $x_{N-1} = y_{N-1}$ by definition. Since the two permutations are adjacent, which means that we can change one into the other by switching two numbers in adjacent positions, those two positions cannot include $z_1 = z_2$. So if we remove the number N from the two permutations A, B, the two shorter permutations are also adjacent. The coordinates of those shorter permutations are $(x_1,x_2,\ldots,x_{N-2})$ and $(y_1,y_2,\ldots,y_{N-2})$. By induction, $\sum_{i=1}^{N-2} |x_i - y_i| = 1$. Since $|x_{N-1} - y_{N-1}| = 0$, we get $\sum_{i=1}^{N-1} |x_i - y_i| = 1$. Case 2: $z_1 \neq z_2$. In this case, since A, B are adjacent, A can be changed into B by switching the z_1-th number and the z_2-th number. Then $|z_1 - z_2| = 1$, and therefore, $|x_{N-1} - y_{N-1}| = 1$, and for any $z \neq z_1, z_2$, we have $x_z = y_z$. So $\sum_{i=1}^{N-1} |x_i - y_i| = 1$. Thus, if the two permutations A and B are adjacent, Condition 1 is true.

If A and B are adjacent, then Condition 2 is also true, for the following simple reason: if the two integers i, j described in Condition 2 exist, then there would be no way to switch a_i and a_j with only one adjacent transposition in order to change A into B. That would be a contradiction.

Now we prove the other direction: if the two conditions are true, then A and B are adjacent. Assume that the two conditions are true. Then, since $\sum_{i=1}^{N-1} |x_i - y_i| = 1$, there are two cases. Case 1: $|x_{N-1} - y_{N-1}| = 1$ and for any $z < N-1$, $x_z = y_z$. In this case, by switching the number N and a number beside it in the permutation A, we can get the permutation B. Hence, the two permutations are adjacent. Case 2: $|x_{N-1} - y_{N-1}| = 0$ and $\sum_{i=1}^{N-2} |x_i - y_i| = 1$. In this case, if we take away the number N from A and B, we get two shorter permutations satisfying the two conditions, so by induction, the two shorter permutations are adjacent. Assume that we can switch the k-th number and the $(k+1)$-th number in the first short permutation to get the second short permutation. For both A and B, since Condition 2 is true, the number N cannot be between those switched numbers. So we can still switch those two numbers as an adjacent transposition to change A into B. Thus A, B are adjacent, and the other direction of the conclusion is also true. ■

Let $L_n = (V_L, E_L)$ denote a $2 \times 3 \times \cdots \times n$ linear array graph. L_n has $n!$ vertices V_L. Each vertex is assigned integer coordinates $(x_1,x_2,\ldots,x_{n-1})$, where $0 \leqslant x_i \leqslant i$ for $1 \leqslant i \leqslant n-1$. The distance between vertices of L_n is the L_1 distance, and two vertices are adjacent (i.e., have an edge between them) if and only if their distance is one.

We now build a bijective map $P : V \to V_L$. Here V is the vertex set of the adjacency graph of permutations $G = (V,E)$. For any $u \in V$ and $v \in V_L$, $P(u) = v$ if and only if u, v have the same coordinates. By Theorem 4, if two permutations are adjacent, their coordinates are adjacent in L_n, and we get:

Theorem 5. *The adjacency graph of permutations is a subgraph* *of the $2 \times 3 \times \cdots \times n$ linear array.*

We show some examples of the embedding in Fig. 1. It can be seen that while each permutation has $n-1$ adjacent permutations, a vertex in the array can have a varied degree from $n-1$ to $2n-3$. Some edges of the array do not exist in the adjacency graph of permutations because they violate condition 2 in Theorem 4.

Proposition 6 *If two vertices are adjacent in the array L_n, their distance in the adjacency graph of permutations, G, is at most $2n-3$, and this bound is tight.*

Proof: Let A and B be two permutations such that X_A and X_B are adjacent in L_n. If they are not adjacent permutations, then they must violate condition 2 in Theorem 4. Without loss of generality, assume

$$A = [a_1, \dots, a_{i-1}, a_i, a_{i+1}, \dots, a_{j-1}, a_j, a_{j+1}, \dots, a_n],$$
$$B = [a_1, \dots, a_{i-1}, a_j, a_{i+1}, \dots, a_{j-1}, a_i, a_{j+1}, \dots, a_n].$$

Clearly, a minimum of $(j-i)+(j-i-1) = 2j-2i-1$ adjacency transpositions are needed to switch a_i and a_j in order to change A into B. When $i=1, j=n$, $2j-2i-1$ reaches the maximum value $2n-3$. Hence, $d(A,B) \leqslant 2n-3$. To see that the bound is tight, consider $A = [1,3,4,\dots,n,2]$ and $B = [2,3,4,\dots,n,1]$. ■

The observation that the permutations' adjacency graph is a subgraph of a linear array shows an approach to design error-correcting rank-modulation codes based on Lee-metric codes. We skip its proof due to its simplicity.

Theorem 7. *Let C be a Lee-metric error-correcting code of length $n-1$, alphabet size no less than n, and minimum distance d. Let C' be the subset of codewords of C that are contained in the array L_n. Then C' is an error-correcting rank-modulation code with minimum distance at least d.*

B. Single-error-correcting Rank-Modulation Code

We now present a family of rank-modulation codes that can correct one error. The code is based on the perfect sphere packing in the Lee-metric space [4]. The code construction is as follows.

Construction 8. (Single-error-correcting rank-modulation code) *Let C_1, C_2 denote two rank-modulation codes constructed as follows. Let A be a general permutation whose coordinates are $(x_1, x_2, \dots, x_{n-1})$. Then A is a codeword in C_1 if and only if the following equation is satisfied:*

$$\sum_{i=1}^{n-1} i x_i \equiv 0 \pmod{2n-1}.$$

A is a codeword in C_2 if and only if the following equation is satisfied:

$$\sum_{i=1}^{n-2} i x_i + (n-1)\cdot(-x_{n-1}) \equiv 0 \pmod{2n-1}.$$

Between C_1 and C_2, choose the code with more codewords as the final output. □

We analyze the code size of Construction 8.

Lemma 9. *The rank-modulation code built in Construction 8 has a minimum cardinality of $\frac{(n-1)!}{2}$.*

Proof: Let $H = (V_H, E_H)$ be a $2 \times 3 \times \cdots \times (n-1) \times (2n-1)$ linear array. Every vertex in H has integer coordinates $(x_1, x_2, \dots, x_{n-1})$, where $0 \leqslant x_i \leqslant i$ for $1 \leqslant i \leqslant n-2$, and $-n+1 \leqslant x_{n-1} \leqslant n-1$.

Given any choice of $(x_1, x_2, \dots, x_{n-2})$ of the coordinates, we would like to see if there is a solution to x_{n-1} (note that $-n+1 \leqslant x_{n-1} \leqslant n-1$) that satisfies the following equation:

$$\sum_{i=1}^{n-1} i x_i \equiv 0 \pmod{2n-1}.$$

Since $\sum_{i=1}^{n-1} i x_i = (n-1)x_{n-1} + \sum_{i=1}^{n-2} i x_i$, and $n-1$ and $2n-1$ are co-prime integers, there is exactly one solution to x_{n-1} that satisfies the above equation. If $x_{n-1} \geqslant 0$, clearly $(x_1, x_2, \dots, x_{n-1})$ are the coordinates of a codeword in the code C_1. If $x_{n-1} \leqslant 0$, then $\sum_{i=1}^{n-2} i x_i + (n-1)\cdot[-(-x_{n-1})] \equiv 0 \pmod{2n-1}$, so $(x_1, x_2, \dots, x_{n-2}, -x_{n-1})$ are the coordinates of a codeword in the code C_2.

Since $0 \leqslant x_i \leqslant i$ for $1 \leqslant i \leqslant n-2$, there are $(n-1)!$ ways to choose $x_1, x_2, \dots, x_{n-2}$. Each choice generates a codeword that belongs either to C_1 or C_2. Therefore, at least one of C_1 and C_2 has cardinality no less than $\frac{(n-1)!}{2}$. ■

Lemma 10. *The rank-modulation code built in Construction 8 can correct one error.*

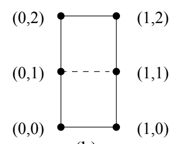
Proof: It has been shown in [4] that for an infinite k-dimensional array, vertices whose coordinates $(x_1, x_2, \dots, x_k)$ satisfy the condition $\sum_{i=1}^{k} i x_i \equiv 0 \pmod{2k+1}$ have a minimum L_1 distance of 3. Let $k = n-1$. Note that in Construction 8, the codewords of C_1 are a subset of the above vertices, while the codewords in C_2 are a subset of the mirrored image of the above vertices, where the last coordinate x_{n-1} is mapped to $-x_{n-1}$. Since the permutations' adjacency graph is a subgraph of the array, the minimum distance of C_1 and C_2 is at least 3. Hence, the code built in Construction 8 can correct one error. ■

Theorem 11. *The code built in Construction 8 is a single-error-correcting rank-modulation code whose cardinality is at least half of optimal.*

Proof: Every permutation has $n-1$ adjacent permutations, so the size of a radius-1 ball, $|\mathcal{B}_1|$, is n. By the sphere packing bound, a single-error-correcting rank-modulation code can have at most $\frac{n!}{n} = (n-1)!$ codewords. The code in Construction 8 has at least $(n-1)!/2$ codewords. ■

Permutation	Coordinates
1 2 3	(0,0)
1 3 2	(0,1)
2 1 3	(1,0)
2 3 1	(1,1)
3 1 2	(0,2)
3 2 1	(1,2)

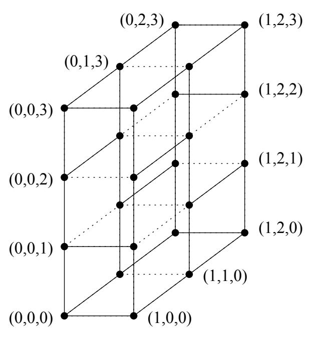
(a)



(b)

Permutation	Coordinates	Permutation	Coordinates
1 2 3 4	(0,0,0)	3 1 2 4	(0,2,0)
1 2 4 3	(0,0,1)	3 1 4 2	(0,2,1)
1 3 2 4	(0,1,0)	3 2 1 4	(1,2,0)
1 3 4 2	(0,1,1)	3 2 4 1	(1,2,1)
1 4 2 3	(0,0,2)	3 4 1 2	(0,2,2)
1 4 3 2	(0,1,2)	3 4 2 1	(1,2,2)
2 1 3 4	(1,0,0)	4 1 2 3	(0,0,3)
2 1 4 3	(1,0,1)	4 1 3 2	(0,1,3)
2 3 1 4	(1,1,0)	4 2 1 3	(1,0,3)
2 3 4 1	(1,1,1)	4 2 3 1	(1,1,3)
2 4 1 3	(1,0,2)	4 3 1 2	(0,2,3)
2 4 3 1	(1,1,2)	4 3 2 1	(1,2,3)

(c)



(d)

Figure 1. Coordinates of permutations, and embedding the adjacency graph of permutations, G, in the $2 \times 3 \times \cdots \times n$ array, L_n. In the two arrays, the solid lines are the edges in both G and L_n, and the dotted lines are the edges only in L_n. (a) Coordinates of permutations for $n = 3$. (b) Embedding G in L_n for $n = 3$. (c) Coordinates of permutations for $n = 4$. (d) Embedding G in L_n for $n = 4$.

V. More Codes and Bounds

It has been shown that the single-error-correcting code built by Construction 8 has a size within half of optimal. There exist code constructions that can build larger codes in many cases. We report here some error-correcting codes built using ad hoc constructions, and compare them with the sphere-packing upper bound and the half-optimal code:

- When $n = 3$, a single-error-correcting code with two codewords – $[1, 2, 3]$ and $[3, 2, 1]$ – can be easily found. The same code is built by Construction 8, and the size meets the sphere-packing upper bound.
- When $n = 4$, an ad hoc construction generates a single-error-correcting code with five codewords: $[1, 2, 4, 3]$, $[3, 1, 4, 2]$, $[3, 2, 4, 1]$, $[4, 1, 3, 2]$ and $[4, 2, 3, 1]$. The code output by Construction 8 has size 4. The sphere-packing bound is 6. It can be shown that the code of size 5 is optimal.
- When $n = 5, 6, 7$, an ad hoc construction generates single-error-correcting codes with 18, 90, and 526 codewords, respectively. The codes output by Construction 8 have size 14, 66, and 388, respectively. The sphere-packing upper bound is 24, 120, and 720, respectively.
- When $n = 5, 6, 7$, there exist two-error-correcting codes of size 6, 23, and 110, three-error-correcting codes of size 2, 10, 34, and four error-correcting codes of size 2, 4, and 14, respectively. All the above codes have a size that is at least one half of the optimal size.

VI. Conclusion

In this paper, we propose a novel data storage scheme for flash memories, the rank-modulation scheme. It can eliminate cell over-programming and also be more robust to asymmetric errors. A rank-modulation scheme uses a new tool – the permutation of cell ranks – to represent data. Consequently, new error-correcting techniques suitable for permutations are needed. We study the properties associated with error-correcting rank-modulation codes, and show that the permutation adjacency graph, which describes the topology of permutations, is a subgraph of a multi-dimensional linear array. As a result, the error-correcting codes for rank modulation can be designed using Lee-metric codes. We present a family of one-error-correcting codes whose size is within half of the optimal size, and also show the results of some other (more ad hoc) code constructions.

It will be interesting to extend the code construction in this paper to design codes that correct two or more errors, by using new Lee-metric codes or suitable lattice interleavers. The codes can also be improved by a better utilization of the sphere packing in the permutation adjacency graph, which is sparser than the array L_n. Alternative embedding of the permutations, known as *permutohedron*, can be explored [3], [8]. (For example, the permutation adjacency graph for four numbers can be embedded as a truncated octahedron.) In addition, it will be interesting to combine the error-correcting codes with data rewriting schemes as in [5].

References

[1] A. Bandyopadhyay, G. Serrano, and P. Hasler, "Programming analog computational memory elements to 0.2% accuracy over 3.5 decades using a predictive method," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, pp. 2148–2151, 2005.

[2] P. Cappelletti, C. Golla, P. Olivo, and E. Zanoni, *Flash memories*. Kluwer Academic Publishers, 1999.

[3] P. Gaiha and S. K. Gupta, "Adjacent vertices on a permutohedron," in *SIAM J. Appl. Math.*, vol. 32, no. 2, pp. 323-327, 1977.

[4] S. W. Golomb and L. R. Welch, "Perfect codes in the Lee metric and the packing of polyominoes," *SIAM J. Appl. Math.*, vol. 18, no. 2, pp. 302–317, Jan. 1970.

[5] A. Jiang, V. Bohossian, and J. Bruck, "Floating codes for joint information storage in write asymmetric memories," in *Proc. IEEE Int. Symp. Information Theory*, Nice, France, pp. 1166–1170, 2007.

[6] A. Jiang, R. Mateescu, M. Schwartz, and J. Bruck, "Rank modulation for flash memories," in *Proc. IEEE Int. Symp. Informationo Theory*, 2008.

[7] M. Kendall and J. D. Gibbons, *Rank correlation methods*. Oxford University Press, NY, 1990.

[8] D. E. Knuth, *The art of computer programming*, vol. 3, 2nd Ed., Addison-Wesley, 1998.

[9] D. H. Lehmer, "Teaching combinatorial tricks to a computer," in *Proc. Sympos. Appl. Math. Combinatorial Analysis*, vol. 10, Amer. Math. Soc., Providence, R.I., pp. 179-193, 1960.