Burst-Deletion-Correcting Codes for Permutations and Multipermutations

Yeow Meng Chee, *Senior Member, IEEE*, San Ling, Tuan Thanh Nguyen, Van Khu Vu, Hengjia Wei, and Xiande Zhang

***Abstract*— Permutation codes and multipermutation codes are widely studied due to various applications in information theory. Designing codes correcting deletion errors has been the main subject of works in the literature and to the best of our knowledge, there exist only optimal codes capable of correcting a single deletion in a permutation. In this paper, we construct several classes of permutation and multipermutation codes that are capable of correcting a burst deletion of length $s \geq 2$, for both stable and unstable models. Efficient error decoders are provided to show the correctness of our constructions.**

***Index Terms*— Error-correction codes, flash memories, permutation codes, multipermutation codes.**

I. Introduction

IN THE 1960's, Slepian [1] proposed using permutation codes and multipermutation codes for transmission in the presence of additive white Gaussian noise. In 2000, permutation codes were suggested as a solution to combat impulse noise and permanent frequency noise in power grids by Vinck [2]. Recently, permutation codes have attracted attention due to their emerging applications in storage systems, such as flash memories [3]–[6].

Flash memories have become a mature technology for the nonvolatile storage of information that can be electrically erased and/or reprogrammed. They are increasingly replacing hard disks, offering the advantage of speed, noise, power consumption, and physical reliability. Flash memories store information in arrays of memory cells, and each cell can store one or more bits of information. The level of a cell can be increased by injecting charge into the cell, and decreased by removing charge from the cell. While adding charge to a single cell can be done fast and simple, removing charge from a single cell is very difficult. It follows from the fact that most flash memory technologies do not allow a single cell to be erased. In order to decrease any cell's level, the whole block needs to be erased (which means to remove the charge from all the cells of the block) and then be reprogrammed. There are two main challenges in flash memories:

(i) **Programming cells**. It is not easy to program each cell exactly to its designated level. In fact, the process of writing a specific level on a cell is designed to cautiously approach the target level from below so as to avoid undesired block erasures in case of overshooting. In order to combat this kind of errors, Jiang *et al.* [3] proposed a *permutation coded rank modulation scheme*. In this setup, the information is stored in the form of ranking of the cells' charges rather than in term of the absolute values of the charges. More specifically, information is written in blocks of n cells, and each block stores a permutation of length n. For example, the relative values of the charge levels in Figure 1 give the permutation $(1, 2, 3, 6, 4, 5)$. This simple coding framework may eliminate the problem of cell block erasures as well as potential cell overshooting issues [4]. For instance, while all absolute values are prone to errors caused by charge leakage, the relative ordering of the quantitative data may remain unchanged. After the work of [3], the permutation coded rank modulation scheme has been extended to tolerate other kinds of errors [4], [5], [7], [8].

(ii) **Memory endurance in aging device**. As mentioned above, it is costly to decrease the cell's level as the whole block needs to be erased and then be reprogrammed. Therefore, it is highly desirable to minimize the number of block erasures. En Gad *et al.* [9] focused on the advantages of multipermutations and claimed that the flexibility could result in better performance. The idea is still using relative levels of the cells instead of the absolute levels, but allowing multiple cells to be in the same relative level. For example, based on the relative levels of cells in Figure 1, we may partition them into three groups, where charge level one includes two cell at the lowest level, charge level two includes those at the middle level, and charge level three includes two cells

Manuscript received October 26, 2018; revised July 21, 2019; accepted July 22, 2019. Date of publication August 7, 2019; date of current version January 20, 2020. S. Ling, T. T. Nguyen, and V. K. Vu were supported by the NTU Grant under Grant M4080456. X. Zhang was supported in part by the NSFC under Grant 11771419 and in part by the Fundamental Research Funds for the Central Universities. This article was presented in part at the 2015 and 2017 Proceedings of the IEEE International Symposium on Information Theory [19], [20].

Y. M. Chee is with the Department of Industrial Systems Engineering and Management, National University of Singapore, Singapore 119077 (e-mail: pvocym@nus.edu.sg).

S. Ling, T. T. Nguyen, and V. K. Vu are with the School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore 639798 (e-mail: lingsan@ntu.edu.sg; nguyentu001@ntu.edu.sgl; vankhu001@ntu.edu.sg).

H. Wei is with the Department of Electrical and Computer Engineering, Ben Gurion University of the Negev, Beersheba 8410501, Israel (e-mail: hjwei05@gmail.com).

X. Zhang is with the Division of Mathematical Sciences, University of Science and Technology of China, Hefei 230026, China (e-mail: drzhangx@ustc.edu.cn).

Communicated by M. Schwartz, Associate Editor for Coding Theory.

Digital Object Identifier 10.1109/TIT.2019.2933819

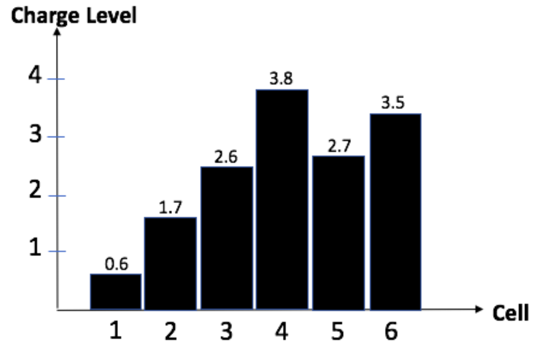

Fig. 1. Charge levels in a block of six cells in a flash memory.

at the highest level. The information is then stored as a multipermutation $(1, 1, 2, 3, 2, 3)$. We refer this model as *multipermutation coded rank modulation scheme*.

Recently, Gabrys *et al*. [11] studied deletion errors in permutation coded rank modulation scheme and Sala *et al*. [12] investigated deletion errors in multipermutation coded rank modulation scheme. Such errors occur when cells are corrupted and the charge levels cannot be read correctly. They provided constructions for permutation codes and multipermutation codes those are capable of correcting a single deletion. Deletions in permutations were classified in [11] and this classification was further extended to multipermutations by Sala *et al*. [12]. In general, deletions are classified according to how they affect the information stored in the remaining cells:

(i) *Stable deletion*: In such a deletion, the absolute values of the cells are known.
(ii) *Unstable deletion*: In such a deletion, only the relative values of the remaining cells are known.

Gabrys *et al*. [11] demonstrated that the models of unstable erasures and stable deletions are equivalent in rank modulation schemes. Here, the former (unstable erasures) refers to the event where the charge levels of certain cells are corrupted and we know their locations. Since unstable erasures happen in flash memories, we study the equivalent model (stable deletions) in this work. Gabrys *et al*. also demonstrated the importance of unstable deletions models (which they referred as *permutation-invariant*) in rank modulation schemes. The scenario is when deletion errors occur, the reader might only distinguish the difference in ranking of the remaining cells rather than their absolute levels. Therefore, the resultant vector is a permutation. This model was further extended to multipermutations by Sala *et al*. [12].

Example 1. Suppose a deletion occurs in the third cell of the block $(1, 2, 3, 6, 4, 5)$ depicted in Fig. 1. In a stable deletion, the remaining components of the permutation give the vector $(1, 2, 6, 4, 5)$, whereas in an unstable deletion, the remaining components of the permutation give the vector $(1, 2, 5, 3, 4)$.

A permutation or multipermutation that experiences unstable deletions loses all information on values and locations of the corrupted cells. Hence, codes capable of correcting unstable deletions need more constraints than codes for correcting stable deletions.

Levenshtein [10] used *Varshamov-Tenengolts (VT) codes* to constructed perfect permutation codes against a stable deletion. Recently, Gabrys *et al*. [11] also used VT codes to construct asymptotically optimal codes against an unstable deletion. In multipermutation, while it is trivial to extend the technique in permutation to construct asymptotically optimal multipermutation codes correcting a stable deletion, designing codes correcting an unstable deletion is still a challenging problem. Sala *et al.* [12] provided a method to construct multipermutation codes against an unstable deletion by using interleaved codes and VT codes. The research on codes correcting multiple deletions is still very limited.

In this paper, we study the related problem of *burst deletions* in rank modulated flash memories, that is, a series of deletions that occur in consecutive cells. The motivation behind considering burst deletions is that as flash memory scales, the parasitic capacitance of adjacent cells increases, which can cause corruptions in a cell to bleed to adjacent cells, through capacitative coupling [13], [14]. Recently, Han *et al*. [15] presented constructions of codes correcting a burst of stable deletions for permutations and multipermutations. Their constructions are mainly based on interleaving technique, and the redundancy of constructed codes grows linearly in term of the length of codewords. In this work, we show that we can do much better by improving the redundancy to be at most $2 \log n + O(1)$, where n is the length of permutations, for codes correcting a burst of s stable deletions, where s is any constant. Our technique can be extended to construct multipermutation codes correcting a burst of stable deletions with at most $2 \log n + O(1)$ redundancy. In both models, the gap between our redundancy and optimality is only $\log n + O(1)$. In addition, we construct the first class of permutation codes that are capable of correcting a burst of s unstable deletions. Motivated by the work of Sala *et al*. [12], we also study multipermutation codes correcting a burst of s unstable deletions. Our results include bounds on the size of optimal codes, code constructions, and efficient decoding algorithms.

The paper is organized as follows. We present notations used throughout the paper in Section II and review the previous works in Section III. In Section IV and Section V, we study stable deletions and present permutation codes and multipermutation codes respectively. Section VI and Section VII are devoted to unstable deletions. Finally, we conclude the paper in section VIII with a summary of the main results.

II. Definitions and Notations

In this section, we give necessary definitions and notations.

A. Permutation Codes

For integers $a \leq b$, $[a, b]$ denotes the set $\{a, a+1, a+2, \ldots, b\}$. Let n be a positive integer and $\mathcal{S}_n$ be the set of all permutations on the set $[1, n]$.

For a permutation $\pi = (\pi_1, \pi_2, \ldots, \pi_n) \in \mathcal{S}_n$ and a set of positions $I \subseteq [1, n]$, define $\pi(I) = \{\pi_i : i \in I\}$. Let X be a set of numbers. For a number a, we define $a(X) = a - |\{x \in X : x < a\}|$.

Example 2. if $\pi = (5, 1, 4, 2, 6, 3)$ and $I = \{3, 4\}$, then $\pi(I) = \{2, 4\}$, $5(\pi(I)) = 5 - 2 = 3$, $1(\pi(I)) = 1 - 0 = 1$, and $6(\pi(I)) = 6 - 2 = 4$.

We first define stable and unstable deletions in permutations.

Definition 1. Assume that $\pi = (\pi_1, \pi_2, \ldots, \pi_n) \in \mathcal{S}_n$ and $I = [i+1, i+s] \subseteq [1, n]$ is a set of size s. We say the permutation π suffers *a burst of s stable deletions* in I, if it results in the vector $(\pi_1, \ldots, \pi_i, \pi_{i+s+1}, \ldots, \pi_n)$. We use π_I to denote this resultant vector. In general, we use $\mathcal{B}_s(\pi)$ to denote the set of all vectors of length $n - s$ received as a result of a burst of s stable deletions in π.

Definition 2. Let π and I be defined as above. We say that π suffers *a burst of s unstable deletions* in I, resulting in the permutation $(\tilde{\pi}_1, \ldots, \tilde{\pi}_{n-s}) \in \mathcal{S}_{n-s}$, if $\tilde{\pi}_j = \pi_j(\pi(I))$ for $1 \le j \le i$ and $\tilde{\pi}_j = \pi_{j+s}(\pi(I))$ for $i+1 \le j \le n-s$. We use $\pi_{\downarrow,I}$ to denote this resultant permutation, and use $\mathcal{B}_{\downarrow,s}(\pi)$ to denote the set of all permutations in $\mathcal{S}_{n-s}$ received as a result of a burst of s unstable deletions in π.

Example 3. Let $\pi = (2, 1, 3, 5, 6, 4)$ and $I = \{3, 4\}$. If π suffers a burst of stable deletions in I, we have the resultant vector $\pi_I = (2, 1, 6, 4)$. On the other hand, if π suffers a burst of unstable deletions in I, then the resultant vector is a permutation, which is $\pi_{\downarrow,I} = (2, 1, 4, 3)$.

Definition 3. A permutation code $\mathcal{C} \subseteq \mathcal{S}_n$ is called an *s-PBSD (or s-PBUD)* code if it can correct a burst of exactly s stable (or unstable) deletions, that is, for distinct $\pi_1, \pi_2 \in \mathcal{C}$, we should have $\mathcal{B}_s(\pi_1) \cap \mathcal{B}_s(\pi_2) = \emptyset$ (or $\mathcal{B}_{\downarrow,s}(\pi_1) \cap \mathcal{B}_{\downarrow,s}(\pi_2) = \emptyset$). Similarly, $\mathcal{C}$ is called an *$\le$s-PBSD (or $\le$s-PBUD)* code if it can correct a burst of at most s stable (or unstable) deletions, that is, for distinct $\pi_1, \pi_2 \in \mathcal{C}$, we should have $\mathcal{B}_k(\pi_1) \cap \mathcal{B}_k(\pi_2) = \emptyset$ (or $\mathcal{B}_{\downarrow,k}(\pi_1) \cap \mathcal{B}_{\downarrow,k}(\pi_2) = \emptyset$) for any $k \le s$.

B. Multipermutation Codes

For a positive integer n and a vector $\mathbf{r} = (r_1, r_2, \ldots, r_m)$ with $n = \sum_{i=1}^m r_i$ and $r_i \ge 1$, which we call *multiplicity vector*, we use $\mathcal{M}(n, \mathbf{r})$ to denote the multiset

$$\{\underbrace{1, \ldots, 1}_{r_1}, \underbrace{2, \ldots, 2}_{r_2}, \ldots, \underbrace{m, \ldots, m}_{r_m}\}.$$

A *multipermutation* $\sigma = (\sigma_1, \ldots, \sigma_n)$ is an arrangement of the elements of that multiset. For example, $(3, 3, 2, 1, 2, 1, 2, 1)$ is a multipermutation of $\{1, 1, 1, 2, 2, 2, 3, 3\}$. We denote the set of all multipermutations on the multiset $\mathcal{M}(n, \mathbf{r})$ as $\mathcal{S}_n^{\mathbf{r}}$. If $r_1 = r_2 = \cdots = r_m = r$, the multiset and the corresponding multipermutations are called *regular*. For simplicity, in this paper, we only discuss regular multipermutations. Our results for regular multipermutations can be easily generalized to non-regular multipermutations.

As we see from Figure 1, each possible ranking of cell charge levels (over a block of cells) yields a single permutation. In multipermutation coded rank modulation scheme, such permutation will be encoded as a multipermutation based on a multiplicity vector $\mathbf{r}$. For example, the permutation $(5, 3, 2, 6, 4, 1)$ is encoded as multipermutation $(3, 2, 1, 3, 2, 1)$ when the multiplicity vector is $\mathbf{r} = (2, 2, 2)$. In general, we derive the multipermutation as follows.

Definition 4. Given a permutation $\pi \in \mathcal{S}_n$, and a multiplicity vector $\mathbf{r} = (r_1, \ldots, r_m)$, where $n = \sum_{i=1}^m r_i$, the *multipermutation representing π based on* $\mathbf{r}$, denoted by $\mathbf{m}_\pi^{\mathbf{r}}$, is derived in the following way:

$$\mathbf{m}_\pi^{\mathbf{r}}(i) = j \text{ if } \sum_{k=1}^{j-1} r_k + 1 \le \pi(i) \le \sum_{k=1}^{j} r_k, i \in [n].$$

When the multipermutation is regular, that is $r_1 = \cdots = r_m$, we have

$$\mathbf{m}_\pi^{\mathbf{r}}(i) = j \text{ if } (j-1)r + 1 \le \pi(i) \le jr, i \in [n].$$

There are several permutations that can be represented by a particular multipermutation. For example, besides the permutation $(5, 3, 2, 6, 4, 1)$, the permutation $(6, 4, 2, 5, 3, 1)$ can also be represented by multipermutation $(3, 2, 1, 3, 2, 1)$ based on the multiplicity vector $\mathbf{r} = (2, 2, 2)$. In fact, if the multiplicity vector $\mathbf{r} = (r_1, r_2, \ldots, r_m)$, there are overall $r_1!r_2!\cdots r_m!$ such permutations. For a multipermutation $\sigma \in \mathcal{S}_n^{\mathbf{r}}$, we use $\mathcal{R}_{\mathbf{r}}(\sigma)$ to denote the set of permutations that can be represented by σ based on the multiplicity vector $\mathbf{r}$, mathematically

$$\mathcal{R}_{\mathbf{r}}(\sigma) = \{\pi \in \mathcal{S}_n : \mathbf{m}_\pi^{\mathbf{r}} = \sigma\}.$$

In multipermutation coded rank scheme, we assume that the storage system works directly on multipermutations. In such a scheme, data is encoded as multipermutations and the decoder will read out the multipermutation resulting from a block of cells, regardless of the underlying permutation. Next, we define a burst of stable deletions and unstable deletions in multipermutations.

Definition 5. Given a positive integer n and a multiplicity vector $\mathbf{r} = (r_1, r_2, \ldots, r_m)$, where $n = \sum_{i=1}^m r_i$, assume that $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n) \in \mathcal{S}_n^{\mathbf{r}}$ and $I = [i+1, i+s] \subseteq [1, n]$ is a set of size s. We say the multipermutation σ suffers *a burst of s stable deletions* in I, if it results in the vector $\sigma_I = (\sigma_1, \ldots, \sigma_i, \sigma_{i+s+1}, \ldots, \sigma_n)$. Let $\mathcal{B}_s(\sigma)$ denote the set of all vectors of length $n - s$ received as a result of a burst of s stable deletions in σ.

For example, let $\sigma = (3, 3, 1, 2, 2, 1, 2, 1) \in \mathcal{S}_8^{(3,3,2)}$ and $I = \{2, 3, 4\}$. If σ suffers a burst of stable deletions in I, then it results in $\sigma_I = (3, 2, 1, 2, 1)$.

Now we define a burst of unstable deletions in multipermutations.

Definition 6. Suppose we have two multipermutations, $\sigma \in \mathcal{S}_n^{\mathbf{r}}$ and $\tilde{\sigma} \in \mathcal{S}_{n-s}^{\tilde{\mathbf{r}}}$, where $\tilde{\mathbf{r}} = (\tilde{r}_1, \ldots, \tilde{r}_m)$ with $0 \le \tilde{r}_i \le r_i$ for $i \in [1, m]$. Let $I = [i+1, i+s] \subseteq [1, n]$. Then we say σ suffers *a burst of s unstable deletions in I and results in $\tilde{\sigma}$* if there exists $\pi \in \mathcal{R}_{\mathbf{r}}(\sigma)$ and $\tilde{\pi} \in \mathcal{R}_{\tilde{\mathbf{r}}}(\tilde{\sigma})$ such that $\tilde{\pi} = \pi_{\downarrow,I}$.

Example 4. Let $n = 6$, $\mathbf{r} = (2, 2, 2)$, and assume that the multipermutation $\sigma = (2, 2, 1, 1, 3, 3) \in \mathcal{S}_6^{\mathbf{r}}$ has a burst of two unstable deletions in $I = \{2, 3\}$. Then σ may result in

$\tilde{\sigma}' = (2,1,3,2)$ since $\pi' = (3,4,1,2,6,5) \in \mathcal{R}_{\mathbf{r}}(\sigma)$, $\tilde{\pi}' = (2,1,4,3) \in \mathcal{R}_{(1,2,1)}(\tilde{\sigma}')$ and $\tilde{\pi}' = \pi'_{\downarrow,I}$; by definition, σ may also result in $\tilde{\sigma}'' = (2,2,3,3)$ since $\pi'' = (3,4,2,1,5,6) \in \mathcal{R}_{\mathbf{r}}(\sigma)$, $\tilde{\pi}' = (2,1,3,4) \in \mathcal{R}_{(0,2,2)}(\tilde{\sigma}'')$ and $\tilde{\pi}'' = \pi''_{\downarrow,I}$. In fact, we may have seven possible vectors as below.

$$\sigma' \in \{(1,1,3,3),(2,1,3,3),(1,1,2,2),\\ (2,1,2,3),(2,1,3,2),(1,1,2,3),(1,1,3,2)\}.$$

From Example 4, we see that there may be more than one possible resulting vectors $\tilde{\sigma}$ received from the unstable deletions of σ in the same set of positions, since there are many choices of π, $\tilde{\pi}$ and $\tilde{\mathbf{r}}$. Given a multiplicity vector $\mathbf{r} = (r_1, \ldots, r_m)$, let $\Gamma_s(\mathbf{r})$ be the set of all possible vectors $\tilde{\mathbf{r}} = (\tilde{r}_1, \ldots, \tilde{r}_m)$ satisfying $0 \le \tilde{r}_i \le r_i$ for $1 \le i \le m$ and $\sum_{i=1}^m \tilde{r}_i = n - s$. Then the set of all possible resulting vectors received from a burst of s unstable deletions of σ, denoted by $\mathcal{B}_{\downarrow,s}(\sigma)$, can be formulated as follows:

$$\mathcal{B}_{\downarrow,s}(\sigma) = \bigcup_{\tilde{\mathbf{r}} \in \Gamma_s(\mathbf{r})} \{\mathbf{m}_{\tilde{\pi}}^{\tilde{\mathbf{r}}} : \tilde{\pi} \in \mathcal{B}_{\downarrow,s}(\pi) \text{ for some } \pi \in \mathcal{R}_{\mathbf{r}}(\sigma)\}.$$

We now have a formal definition for multipermutation codes correcting a burst of stable (or unstable) deletions.

Definition 7. A multipermutation code $\mathcal{C} \subseteq \mathcal{S}_n^{\mathbf{r}}$ is called an *s-MBSD (or s-MBUD)* code if it can correct a burst of exactly s stable (or unstable) deletions. In other words, for distinct $\sigma_1, \sigma_2 \in \mathcal{C}$, we have $\mathcal{B}_s(\sigma_1) \cap \mathcal{B}_s(\sigma_2) = \emptyset$ (or $\mathcal{B}_{\downarrow,s}(\sigma_1) \cap \mathcal{B}_{\downarrow,s}(\sigma_2) = \emptyset$). Similarly, $\mathcal{C}$ is called an *≤s-MBSD (or ≤s-MBUD)* code if it can correct a burst of at most s stable (or unstable) deletions. In other words, for distinct $\sigma_1, \sigma_2 \in \mathcal{C}$, we have $\mathcal{B}_k(\sigma_1) \cap \mathcal{B}_k(\sigma_2) = \emptyset$ (or $\mathcal{B}_{\downarrow,k}(\sigma_1) \cap \mathcal{B}_{\downarrow,k}(\sigma_2) = \emptyset$) for any $k \le s$.

Throughout this paper, when $n = st$ for appropriate integers s and t, a permutation (or multipermutation) $\mathbf{c} = (c_1, c_2, \ldots, c_n)$ is written as an array

$$\begin{bmatrix} c_1 & c_{s+1} & \cdots & c_{(j-1)s+1} & \cdots & c_{(t-1)s+1} \\ c_2 & c_{s+2} & \cdots & c_{(j-1)s+2} & \cdots & c_{(t-1)s+2} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ c_s & c_{2s} & \cdots & c_{js} & \cdots & c_n \end{bmatrix}.$$

Hence, we may speak of an $s \times t$ permutation (or multipermutation), and refer to rows or columns of an $s \times t$ permutation (or multipermutation). The ith row of an $s \times t$ permutation (or multipermutation) $\mathbf{c}$ is denoted by

$$\mathbf{c}_{(s,i)} = (c_i, c_{s+i}, \ldots, c_{n-s+i}), \text{ and}$$

the jth column of $\mathbf{c}$ is denoted by

$$\mathbf{c}^{(s,j)} = (c_{(j-1)s+1}, c_{(j-1)s+2}, \ldots, c_{js}).$$

Observation 1. A burst of length s deletions in an $s \times t$ permutation (or similarly multipermutation) deletes exactly one symbol from each row and affects at most two adjacent columns.

Definition 8. For two vectors $\mathbf{u} = (u_1, u_2, \ldots, u_m)$ and $\mathbf{v} = (v_1, v_2, \ldots, v_n)$, the *concatenation* of $\mathbf{u}$ and $\mathbf{v}$ is the vector $\mathbf{u}||\mathbf{v} = (u_1, u_2, \ldots, u_m, v_1, v_2, \ldots . v_n)$.

III. Previous Works

We now review the previous works on deletion-correcting codes. Deletions in q-ary codes were defined in [16], [17]. In such models, the deletions do not affect the value of remaining symbols. Therefore, we refer deletions in q-ary as stable deletions.

A. To Combat a Single Stable Deletion

For a positive integer $a \in \mathbb{Z}_{n+1}$, let

$$\mathcal{C}_a(n) = \left\{ \mathbf{u} \in \{0,1\}^n : \sum_{i=1}^n i u_i \equiv a \pmod{n+1} \right\},$$

where u_i is the ith component of $\mathbf{u}$. Then $\mathcal{C}_a(n)$ are the family of binary codes known as the *Varshamov-Tenengolts (VT) codes* [16]. These codes are capable of correcting a single stable deletion. It is known that the choice $a = 0$ maximizes the cardinality of the codes.

As in [17], the *signature* of $\mathbf{u} = (u_1, u_2, \ldots, u_n) \in [1,q]^n$ is the binary vector $\alpha(\mathbf{u}) = (\alpha(u_1), \ldots, \alpha(u_{n-1}))$ of length $n-1$, where $\alpha(u_i) = 1$ if $u_{i+1} \ge u_i$, and 0 otherwise, for all $i \in [1, n-1]$. For example, if $\mathbf{u} = (1,2,1,3,2,3)$, then $\alpha(\mathbf{u}) = (1,0,1,0,1)$. Levenshteĭn [10] showed that

$$\mathcal{P}_a(n) = \{\pi \in \mathcal{S}_n : \alpha(\pi) \in \mathcal{C}_a(n-1)\}$$

is a permutation code that is capable of correcting a single stable deletion. Since the union of these codes over $a \in \mathbb{Z}_n$ is exactly $\mathcal{S}_n$, one of them has size at least $n!/n = (n-1)!$.

Tenengolts [17] also generalized the binary VT codes to nonbinary ones. For any $a \in \mathbb{Z}_n$ and $b \in \mathbb{Z}_q$, let

$$\mathcal{C}_{a,b}(n;q) = \left\{ \mathbf{u} \in [1,q]^n : \alpha(\mathbf{u}) \in \mathcal{C}_a(n-1) \right.\\ \left. \text{and } \sum_{i=1}^n u_i \equiv b \pmod q \right\}.$$

The codes $\mathcal{C}_{a,b}(n;q)$ are capable of correcting a single stable deletion, and one of them has size at least q^{n-1}/n.

B. To Combat a Single Unstable Deletion

For a permutation $\pi = (\pi_1, \pi_2, \ldots, \pi_n) \in \mathcal{S}_n$, its *inverse* is the permutation $\pi^{-1} = (\pi_1^{-1}, \pi_2^{-1}, \ldots, \pi_n^{-1})$, where π_i^{-1} is the location of i in π. Gabrys *et al.* [11] constructed asymptotically optimal permutation codes that can correct a single unstable deletion as follows. Given $a, b \in \mathbb{Z}_n$, let

$$\mathcal{C}_{a,b}(n) = \left\{ \pi \in \mathcal{S}_n : \alpha(\pi) \in \mathcal{C}_a(n-1) \text{ and } \alpha(\pi^{-1}) \in \mathcal{C}_b(n-1) \right\}.$$

An efficient decoding algorithm was given by Gabrys *et al.* [11].

Sala *et al.* [12] provided a method to correct unstable deletion in multipermutations by using interleaved codes and VT codes. They focused on regular multipermutations, i.e. $r_1 = r_2 = \cdots = r_m = r$. Given $a \in \mathbb{Z}_{n+1}$, they defined a code $\mathcal{C}_a^r(n)$, which can correct a single unstable deletion, as follows.

$$\mathcal{C}_a^r(n) = \\ \left\{ \sigma = (\sigma_1, \ldots, \sigma_n) \in \mathcal{S}_n^{\mathbf{r}} : \sigma_i \equiv 0 \pmod 2 \text{ for } i \in \left[1, \frac{n}{2}\right], \right.$$

$$\sigma_i \equiv 1 \pmod 2 \text{ for } i \in \left[\frac{n}{2}+1, n\right], \sum_{i=2}^{n} i\alpha(\sigma_{i-1}) + 1 \equiv a$$
$$\pmod{n+1}, \text{ and } \left|\sigma\left(\frac{n}{2}\right) - \sigma\left(\frac{n}{2}+1\right)\right| \geq 3 \bigg\}.$$

C. To Combat a Burst of Stable Deletions

Han *et al.* [15] presented constructions of codes correcting a burst of stable deletions for permutations and multipermutations. Their constructions are mainly based on interleaving technique. Specifically, for code length $n = st$, they partitioned a set of n symbols into s groups each of t symbols, and constructed codes that can correct a stable deletion over the set of t symbols in each group, which is similar to the work of Levenshteĭn [10]. The resultant interleaved code is capable of correcting a burst of s stable deletions. Since in each group the code correcting one stable deletion has size $(t-1)!$, such code has size at least $((t-1)!)^s$.

D. Our Main Idea and Contribution

One may improve this lower bound by applying the technique proposed by Schoeny *et al.* [18], in which a new class of codes was introduced and can correct single deletion if the deletion is within P consecutive positions. For example, in permutations, we take the first row of an $s \times (n/s)$ permutation to be from a nonbinary VT code (so we only take the codewords that have different components). Then the position of the deletion in the first row can be determined exactly. Thus deletions in the remaining $(s-1)$ rows are known to be within two consecutive positions. To recover these deletions, one can set up the constraint for sum of symbols in each row to find the deleted symbols and then use the codes from [18] to determine their positions. However, this incurs huge redundancy when n is large. Actually, we need roughly $2\log n$ bits of redundancy to correct the deletion in the first row and $\log n$ bits for each of the remaining $(s-1)$ rows. Hence, the redundancy is approximately $(s+1)\log n + O(1)$.

In this paper, we provide constructions for permutations and multipermutations, which improve the redundancy from $(s+1)\log n + O(1)$ to at most $2\log n + O(1)$. In our constructions, like before we first use a non-binary VT code to determine the exact location of the deletion in the first row. From Observation 1, the burst of deletions spans at most two adjacent columns, and we can determine their position from the location of the deletion in the first row. Therefore, we only need to determine the overall ordering of the remaining deleted symbols. To overcome this problem, we use two functions p and μ to map every two adjacent columns to a symbol of the set $[1, (2s)!]$ to record the ordering. Then we set up constraints for the sum of these symbols so that one can recover the ordering. Note that the redundancy for this step only depends on s instead of n. Our constructions reduce the total redundancy from $(s+1)\log n + O(1)$ to $2\log n + O(1)$. The permutation projection function p and the permutation rank function μ are defined below.

Definition 9. For an integer vector of length n, $\mathbf{u} = (u_1, u_2, \ldots, u_n)$, the permutation projection of $\mathbf{u}$, denoted by $\mathsf{p}(\mathbf{u})$, is a permutation in $\mathcal{S}_n$ where:

$$\mathsf{p}(\mathbf{u})_i = |\{j : u_j < u_i, 1 \leq j \leq n\}| + |\{j : u_j = u_i, 1 \leq j \leq i\}|.$$

Observe that $\mathsf{p}(\mathbf{u})$ is a permutation that records the ordering of the symbols in $\mathbf{u}$, and it is well-defined for vectors of different or repeated symbols. When $\mathbf{u}$ is a vector of different symbols, i.e. $u_i \neq u_j$ for $i \neq j$, the term $|\{j : u_j = u_i, 1 \leq j \leq i\}| = 0$. On the other hand, when $\mathbf{u}$ contains repeated symbols, we rank the order of such symbols as an increasing order according to their appearances in $\mathbf{u}$.

Example 5. Let $\mathbf{u} = (6, 2, 5, 1, 8, 4)$ and $\mathbf{v} = (1, 2, 4, 5, 8, 10)$. Then $\mathsf{p}(\mathbf{u}) = (5, 2, 4, 1, 6, 3)$ and $\mathsf{p}(\mathbf{v}) = (1, 2, 3, 4, 5, 6)$. Let $\mathbf{u}' = (1, 1, 2, 2, 1, 2)$ and $\mathbf{v}' = (1, 1, 1, 2, 2, 2)$. Then $\mathsf{p}(\mathbf{u}') = (1, 2, 4, 5, 3, 6)$, $\mathsf{p}(\mathbf{v}') = (1, 2, 3, 4, 5, 6)$.

Definition 10. Let $\ell : \mathcal{S}_n \to [1, n!]$ be a bijection such that $\ell(\pi)$ is the lexicographic rank of π in $\mathcal{S}_n$. Let $\mathbf{u} = (u_1, u_2, \ldots, u_n)$ be an integer vector of length n. We define the *permutation rank of* $\mathbf{u}$, denoted by $\mu(\mathbf{u}) = \ell(\mathsf{p}(\mathbf{u})) \in [1, n!]$.

Example 6. Let $\mathbf{u} = (1, 1, 2, 2, 1, 2)$ and $\mathbf{v} = (1, 2, 1, 2, 1, 2)$. The permutation projections of $\mathbf{u}, \mathbf{v}$ are $\mathsf{p}(\mathbf{u}) = (1, 2, 4, 5, 3, 6)$, $\mathsf{p}(\mathbf{v}) = (1, 4, 2, 5, 3, 6)$.

Observation 2. From $\mu(\mathbf{u})$ and the set $\{u_i : 1 \leq i \leq m\}$, we can determine the sequence $\mathbf{u}$ exactly. For example, if $m = 4$ and $\mu(\mathbf{u}) = 2$, then $\mathsf{p}(\mathbf{u}) = (1, 2, 4, 3)$. Furthermore, if $\{u_1, u_2, u_3, u_4\} = \{1, 3, 4, 5\}$, then $\mathbf{u} = (1, 3, 5, 4)$. When $\mathbf{u}$ contains repeated symbols, sometimes we don't have a valid $\mathbf{u}$ given the value $\mathsf{p}(\mathbf{u})$ and the set of symbols.

Example 7. Let $\mu(\mathbf{u}) = 2$ and we know the set of symbols in $\mathbf{u}$ is $\{1, 1, 2, 2, 2, 3\}$. From $\mu(\mathbf{u}) = 2$ and $n = 6$, we have $\mathsf{p}(\mathbf{u}) = (1, 2, 3, 4, 6, 5)$. We then get $\mathbf{u} = (1, 1, 2, 2, 3, 2)$. However, if the set of symbols in $\mathbf{u}$ is $\{1, 1, 2, 2, 3, 3\}$, then there does not exist a vector $\mathbf{u}$ such that $\mathsf{p}(\mathbf{u}) = (1, 2, 3, 4, 6, 5)$.

We now present our construction for permutation codes correcting a single burst of s stable deletions.

IV. Permutation Codes Correcting a Burst of Stable Deletions

A. Upper Bound

Let $A_{\mathsf{PBSD}}(n, s)$ denote the maximum size of an s-PBSD code in $\mathcal{S}_n$. As defined earlier, for a permutation π in $\mathcal{S}_n$, $\mathcal{B}_s(\pi)$ is the set of all vectors of length $n-s$ received as a result of a burst of s stable deletions in π. Let $\mathcal{B}_s(\mathcal{S}_n) = \bigcup_{\pi \in \mathcal{S}_n} \mathcal{B}_s(\pi)$.

A code $\mathcal{C}$ is an s-PBSD permutation code if and only if for distinct $\pi_1, \pi_2 \in \mathcal{C}$, we have $\mathcal{B}_s(\pi_1) \cap \mathcal{B}_s(\pi_2) = \emptyset$. It is easy to see that for each permutation π, we have $|\mathcal{B}_s(\pi)| = n - s + 1$. We also have $|\mathcal{B}_s(\mathcal{S}_n)| = n!/s!$, since $\mathcal{B}_s(\mathcal{S}_n)$ is the set of all sequences consisting of $n-s$ distinct symbols from an alphabet of size n. Consequently, we have the following.

Theorem 1. *Let $n > s$ be positive integers. Then*

$$A_{\mathsf{PBSD}}(n, s) \leq \frac{n!}{s!(n-s+1)}.$$

B. A Construction of s-PBSD Codes

We assume that $n = st$, where $s \geq 2$ and t is even.

Construction 1. *Let $a \in \mathbb{Z}_t$, $b \in \mathbb{Z}_n$, and $c, d \in \mathbb{Z}_{(2s)!}$. Let $\mathcal{C}_s(n; a, b, c, d)$ be the set of all $s \times t$ permutations $\pi \in \mathcal{S}_n$ such that the following holds:*

(i) The first row $\pi_{(s,1)}$ of π satisfies $\pi_{(s,1)} \in \mathcal{C}_{a,b}(t; n)$.
(ii) Pairs of adjacent columns of π satisfy the conditions
(1) $\sum_{j=1}^{t/2} \mu(\pi^{(s,2j-1)}||\pi^{(s,2j)}) \equiv c \pmod{(2s)!}$, and
(2) $\sum_{j=1}^{t/2-1} \mu(\pi^{(s,2j)}||\pi^{(s,2j+1)}) \equiv d \pmod{(2s)!}$.

Theorem 2. *The permutation code $\mathcal{C}_s(n; a, b, c, d)$ from Construction 1 is an s-PBSD code over $\mathcal{S}_n$.*

Proof: We establish the theorem by giving a decoding algorithm for $\mathcal{C}_s(n; a, b, c, d)$ that recovers from a burst of exactly s stable deletions.

Suppose a burst of s stable deletion occurs in a codeword $\pi = (\pi_1, \pi_2, \ldots, \pi_n) \in \mathcal{C}_s(n; a, b, c, d)$, giving $\tilde{\pi} = (\tilde{\pi}_1, \tilde{\pi}_2, \ldots, \tilde{\pi}_{(t-1)s})$. Let $\rho = (\tilde{\pi}_1, \tilde{\pi}_{s+1}, \ldots, \tilde{\pi}_{(t-2)s+1})$, and $\gamma_j = (\tilde{\pi}_{(j-1)s+1}, \tilde{\pi}_{(j-1)s+2}, \ldots, \tilde{\pi}_{js})$, for $1 \leq j \leq t-1$. So ρ is the first row of the received codeword and γ_j is the jth column. Since the first row of π is a codeword of a single deletion correcting code by construction, we can recover the first row of π from ρ. From the recovered row, we can also determine exactly where the deletion has taken place since all entries of the row are distinct.

Suppose the deletion in the first row of π occured at position $(j_1 - 1)s + 1$, where $2 \leq j_1 \leq t - 1$. Then we have

(i) $\pi^{(s,j)} = \gamma_j$ for $1 \leq j < j_1 - 1$, and
(ii) $\pi^{(s,j)} = \gamma_{j-1}$ for $j_1 + 1 \leq j \leq t$,

since the s deletions are adjacent and occur across at most two adjacent columns, $\pi^{(s,j_1-1)}$ and $\pi^{(s,j_1)}$, of π. This leaves two columns $\pi^{(s,j_1-1)}$ and $\pi^{(s,j_1)}$ to recover. The symbols appearing in these two columns can be deduced from the other columns as the burst deletion is stable. Knowing the ordering of these symbols in $\pi^{(s,j_1-1)}||\pi^{(s,j_1)}$ would enable us to recover $\pi^{(s,j_1-1)}$ and $\pi^{(s,j_1)}$. This ordering is precisely encoded in $\mu(\pi^{(s,j_1-1)}||\pi^{(s,j_1)})$, which can be computed as follows:

when j_1 is even,

$$\mu(\pi^{(s,j_1-1)}||\pi^{(s,j_1)}) \equiv c - \sum_{j \in [1,t/2] \setminus \{j_1/2\}} \mu(\pi^{(s,2j-1)}||\pi^{(s,2j)}) \pmod{(2s)!} \text{ and}$$

when j_1 is odd and $j_1 \neq 1$,

$$\mu(\pi^{(s,j_1-1)}||\pi^{(s,j_1)}) \equiv d - \sum_{j \in [1,t/2-1] \setminus \{(j_1-1)/2\}} \mu(\pi^{(s,2j)}||\pi^{(s,2j+1)}) \pmod{(2s)!}.$$

When $j_1 = 1$, all the s deletions occur in the first column of π, and can be recovered with knowledge of $\mu(\pi^{(s,1)}||\pi^{(s,2)})$, which can be computed using condition (1). ∎

Corollary 1. *There exists an s-PBSD code of size at least $\dfrac{s}{((2s)!)^2} \cdot \dfrac{n!}{n^2}$.*

Proof: We have $\cup_{a \in \mathbb{Z}_t, b \in \mathbb{Z}_n, c,d \in \mathbb{Z}_{(2s)!}} \mathcal{C}_s(n; a, b, c, d) = \mathcal{S}_n$, where $t = \frac{n}{s}$. There exist a, b, c, and d such that the size of $\mathcal{C}_s(n; a, b, c, d)$ is at least as large as the average. ∎

Remark 1. When s is fixed, our construction forms a family of s-PBSD codes whose size is at least $\Omega(n!/n^2)$ while the upper bound, according to Theorem 1, is $\Theta(n!/n)$. The redundancy of our codes is at most $2 \log n + O(1)$. It is easy to see that the decoding algorithm runs in time linear in the length of permutation. In addition, the constant term in the lower bound from Corollary 1, which is $\dfrac{s}{((2s)!)^2}$, can be improved in several cases. Specifically, when $s = 2$, we modify Construction 1 to obtain a family of codes with size at least $n!/n^2$, instead of $n!/(288n^2)$ (refer to Subsection IV-D, Corollary 3).

C. A Construction of ≤s-PBSD Codes

In this subsection, we consider the problem of correcting a burst of at most s stable deletions. For general s, one can take the intersection of a family of $\mathcal{C}_i(n; a_i, b_i, c_i, d_i)$, where $2 \leq i \leq s$, together with $\mathcal{P}_{a_1}(n)$ to construct an $\leq$s-PBSD permutation code. Such a code has size at least $\dfrac{s!n!}{\prod_{i=2}^{s}((2i)!)^2 \; n^{2s-1}}$. Below, we show that we can do a little better. Assume that $i \mid n$ for all $1 \leq i \leq s$, and $n = 2ts$ for some even integer t.

Construction 2. *Let $c, d \in \mathbb{Z}_{(4s)!}$, and let $\mathbf{a} = (a_i)_{i=1}^s$ and $\mathbf{b} = (b_j)_{j=2}^s$ be two sequences of nonnegative integers such that $a_i \in \mathbb{Z}_{n/i}$, for $i = 1, 2, \ldots, s$, and $b_j \in \mathbb{Z}_n$, for $j = 2, 3, \ldots, s$. Let $\mathcal{C}_s(n; \mathbf{a}, \mathbf{b}, c, d)$ be the set of all permutations $\pi \in \mathcal{S}_n$ such that the following holds:*

(i) $\pi \in \mathcal{P}_{a_1}(n)$.
(ii) When π is viewed as an $i \times (n/i)$ permutation, its first row $\pi_{(i,1)}$ belongs to $\mathcal{C}_{a_i,b_i}(n/i; n)$, for $i = 2, 3, \ldots, s$.
(iii) When viewed as a $2s \times t$ permutation, pairs of adjacent columns of π satisfy the conditions
(1) $\sum_{j=1}^{t/2} \mu(\pi^{(2s,2j-1)}||\pi^{(2s,2j)}) \equiv c \pmod{(4s)!}$, and
(2) $\sum_{j=1}^{t/2-1} \mu(\pi^{(2s,2j)}||\pi^{(2s,2j+1)}) \equiv d \pmod{(4s)!}$.

Theorem 3. *The permutation code $\mathcal{C}_s(n; \mathbf{a}, \mathbf{b}, c, d)$ from Construction 2 is an ≤s-PBSD code.*

Proof: We establish the theorem by giving a decoding algorithm for $\mathcal{C}_s(n; \mathbf{a}, \mathbf{b}, c, d)$ that recovers from a burst of at most s stable deletions.

Suppose a stable burst deletion of length $k \leq s$ occurs in a codeword $\pi = (\pi_1, \pi_2, \ldots, \pi_n) \in \mathcal{C}_s(n; \mathbf{a}, \mathbf{b}, c, d)$, giving $\tilde{\pi} = (\tilde{\pi}_1, \tilde{\pi}_2, \ldots, \tilde{\pi}_{n-k})$. The value of k can be determined from the length of $\tilde{\pi}$.

If $k = 1$, we can recover π since $\pi \in \mathcal{P}_{a_1}(n)$.

If $k \geq 2$, we view π as an $k \times (n/k)$ permutation and recover the first row of π and the location of the deletion as in the proof of Theorem 2. Suppose $\pi_{j_1 k+1}$ (in the first row of π) is deleted. Since the k deletions are adjacent, we necessarily have

(i) $\pi_i = \tilde{\pi}_i$, for $1 \leq i \leq j_1 k - k + 1$; and
(ii) $\pi_i = \tilde{\pi}_{i-k}$, for $j_1 k + k + 1 \leq i \leq n$.

It remains to determine the sequence $\pi_{(j_1-1)k+2}, \pi_{(j_1-1)k+2}, \ldots, \pi_{(j_1+1)k}$ of length $2k-1$. To do this, we view π as a $2s \times (n/2s)$ permutation. Then the sequence to be determined spans at most two adjacent columns of π and can be recovered as in the proof of Theorem 2.

If $j_1 = 0$, the deletions occur in the first k positions. So the sequence to be determined spans the first column of the $2s \times (n/2s)$ permutation π and still can be recovered as in the proof of Theorem 2. ■

Corollary 2. *There exists an* $\leq s$*-PBSD code of size at least* $\dfrac{s!}{((4s)!)^2} \cdot \dfrac{n!}{n^{2s-1}}$.

In general, when s is fixed, our construction forms a family of $\leq s$-PBSD codes whose size is at least $\Omega(n!/n^{2s-1})$. The constant term, which is $\dfrac{s!}{((4s)!)^2}$, can be improved in several cases. In the next subsection, we show the improvement when $s = 2$.

D. A Major Improvement When $s = 2$

Definition 11. For a vector $\mathbf{u} = (u_1, u_2, \ldots, u_n)$, let $\beta(\mathbf{u})$ be the number of inversions in the vector, so

$$\beta(\mathbf{u}) = |(u_i, u_j), i < j : u_i > u_j|.$$

Example 8. Let $\mathbf{u} = (1, 3, 3, 2)$ and $\mathbf{v} = (3, 2, 4, 1)$. We have $\beta(\mathbf{u}) = 2, \beta(\mathbf{v}) = 4$.

Recall that when $s = 2$ and $n = 2t$ for some integer t, a permutation $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ is also written as the array

$$\begin{bmatrix} \pi_1 & \pi_3 & \cdots & \pi_{2i-1} & \cdots & \pi_{n-1} \\ \pi_2 & \pi_4 & \cdots & \pi_{2i} & \cdots & \pi_n \end{bmatrix}.$$

Construction 3. *Let $a \in \mathbb{Z}_t$, $b \in \mathbb{Z}_n$, and $c \in \mathbb{Z}_2$. Let $\mathcal{C}_2(n; a, b, c)$ be the set of all $2 \times t$ permutations $\pi \in \mathcal{S}_n$ such that the following holds:*

(i) The first row $\pi_{(2,1)}$ of π is a codeword (with different components) of nonbinary VT-code as defined in Subsection III-A, that is,

$$\pi_{(2,1)} \in \mathcal{C}_{a,b}(t; n).$$

(ii) The second row $\pi_{(2,2)} = (\pi_2, \pi_4, \ldots, \pi_n)$ of π satisfies the condition

$$\beta(\pi_{(2,2)}) \equiv c \pmod 2$$

Theorem 4. *The permutation code $\mathcal{C}_2(n; a, b, c)$ from Construction 3 is a* 2*-PBSD code over $\mathcal{S}_n$.*

Proof: We establish the theorem by giving a decoding algorithm for $\mathcal{C}_2(n; a, b, c)$ that recovers a burst of two stable deletions.

Suppose that two deletions occur in a codeword $\pi = (\pi_1, \pi_2, \ldots, \pi_n) \in \mathcal{C}_2(n; a, b, c)$, giving $\tilde{\pi} = (\tilde{\pi}_1, \tilde{\pi}_2, \ldots, \tilde{\pi}_{2t-2})$. Let $\rho = (\tilde{\pi}_1, \tilde{\pi}_3, \ldots, \tilde{\pi}_{2t-3})$, and $\gamma_j = (\tilde{\pi}_{2j-1}, \tilde{\pi}_{2j})$, for $1 \leq j \leq t-1$. So ρ is the first row of the received codeword and γ_j is the jth column. Since the first row of π is a codeword of a single deletion correcting code by construction, we can recover the first row of π from ρ. From the recovered row, we can also determine exactly where the deletion has taken place since all entries of the row are distinct.

Suppose the deletion in the first row of π occured at position $(2j_1 - 1)$, in other words, σ_{2j_1-1} was deleted, where $2 \leq j_1 \leq t-1$. Assume that the other deleted symbol is x. Since 2 deletions are adjacent and occur across at most two adjacent columns, $\pi^{(s,j_1-1)}$ and $\pi^{(s,j_1)}$, of π. If $j_1 = 1$, we can conclude that two deletions occurred at the first column and $x = \pi_2$. If $j_1 \geq 2$, we then have

(i) $\pi_i = \tilde{\pi}_i$ for all $1 \leq i \leq 2j_1 - 3$, and
(ii) $\pi_i = \tilde{\pi}_{i-2}$ for all $2j_1 + 1 \leq i \leq n$.

Hence, we have two possible choices for second row of original permutation as follows

(i) $\pi_{2,2} = A \triangleq (\tilde{\pi}_2, \ldots, \tilde{\pi}_{2j_1-4}, \tilde{\pi}_{2j_1-2}, x, \tilde{\pi}_{2j_1}, \ldots, \tilde{\pi}_{2t-2})$, or
(ii) $\pi_{2,2} = B \triangleq (\tilde{\pi}_2, \ldots, \tilde{\pi}_{2j_1-4}, x, \tilde{\pi}_{2j_1-2}, \tilde{\pi}_{2j_1}, \ldots, \tilde{\pi}_{2t-2})$.

The second constraint $\beta(\pi_{(2,2)}) \equiv c \pmod 2$ helps us distinguish these two cases, since the values of $\beta(A)$ and $\beta(B)$ differ by one. ■

Corollary 3. *There exists a* 2*-PBSD code of size at least* $\dfrac{n!}{n^2}$.

Proof: We have $\cup_{a \in \mathbb{Z}_t, b \in \mathbb{Z}_n, c \in \mathbb{Z}_2} \mathcal{C}_2(n; a, b, c) = \mathcal{S}_n$, where $t = n/2$. There exist a, b, and c such that the size of $\mathcal{C}_s(n; a, b, c)$ is at least as large as the average. ■

When $s = 2$, the intersection of $\mathcal{P}_a(n)$ and the code from Corollary 3 gives a ≤ 2-PBSD code.

Theorem 5. *There exists an* ≤ 2*-PBSD code of size at least* $n!/n^3$.

Remark 2. When $s = 2$, from Corollary 1, the 2-PBSD code from Construction 1 has size at least $n!/(288n^2)$. This lower bound is improved to $n!/n^2$ from Corollary 3. Similarly, when $s = 2$, from Corollary 2, the ≤ 2-PBSD code from Construction 2 has size at least $n!/(812851200n^3)$. Theorem 5 improves this lower bound to at least $n!/n^3$.

V. Multipermutation Codes Correcting a Burst of Stable Deletions

A. The Difference Between Permutations and Multipermutations

Recall that to construct permutation codes correcting a burst of s deletions, we write any permutation of length n as an $s \times t$ array and use the fact that a burst of deletions spans at most two columns in this array. We then use the VT constraint (refer to Construction 1) for the first row of the array in order to find the location of deleted symbol in the first row, which consequently helps us identify the two columns where the burst of deletions occurs. However, for multipermutations, due to the multiplicity of the symbols, we can only locate the deletions within at most $r+1$ columns. So we treat the multipermutation as an $s(r+1) \times t$ array and encode the columns. As in Construction 1, we make use of the permutation projection function and the permutation rank function to encode the columns.

B. Code Construction for s-MBSD Codes

Construction 4. *Given $n = rm = (r+1)st$, t even. Let $a \in \mathbb{Z}_{(r+1)t}$, $b \in \mathbb{Z}_m$, and $c, d \in \mathbb{Z}_{(2s(r+1))!}$. Let $\mathcal{C}_s(n; r, a, b, c, d)$ be the set of all regular multipermutations $\sigma \in \mathcal{S}_n^{\mathbf{r}}$ such that the following holds:*

(i) When we view σ as an $s \times (r+1)t$ array, the first row of σ satisfies $\sigma_{(s,1)} \in \mathcal{C}_{a,b}((r+1)t; m)$.

(ii) When we view σ as an $s(r+1) \times t$ array, the pairs of adjacent columns satisfy

(1) $\sum_{i=1}^{t/2} \mu(\sigma^{(s(r+1),2i-1)} || \sigma^{(s(r+1),2i)}) \equiv c \pmod{(2s(r+1))!}$, and

(2) $\sum_{i=1}^{t/2-1} \mu(\sigma^{(s(r+1),2i)} || \sigma^{(s(r+1),2i+1)}) \equiv d \pmod{(2s(r+1))!}$.

Theorem 6. *The multipermutation code $\mathcal{C}_s(n; r, a, b, c, d)$ from Construction 4 is a regular s-MBSD code over $\mathcal{S}_n^{\mathbf{r}}$.*

Proof: We prove the correctness of the theorem by providing a decoding algorithm for $\mathcal{C}_s(n; r, a, b, c, d)$ that recovers from a burst of exactly s stable deletions.

Suppose a burst of s stable deletions occurs in a codeword $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)$ in $\mathcal{C}_s(n; r, a, b, c, d)$, giving $\tilde{\sigma} = (\tilde{\sigma}_1, \tilde{\sigma}_2, \ldots, \tilde{\sigma}_{n-s})$. Let $\rho = (\tilde{\sigma}_1, \tilde{\sigma}_{s+1}, \ldots, \tilde{\sigma}_{((r+1)t-2)s+1})$. So ρ is the first row of the received word when we view it as an $s \times (r+1)t$ array. Since the first row of σ is a codeword of a single-deletion-correcting code by construction, we can recover the first row of σ from ρ. Notice that the deleted symbol belongs to a run of length at most r. From the recovered row, we can determine the run of the deleted symbol. We now view σ as an $s(r+1) \times t$ array and observe that the run of deleted symbols spans at most two adjacent columns of this array. We then use the second constraint to recover the burst of deletions as in the proof of Theorem 2. ■

Corollary 4. *There exists a regular s-MBSD code of size at least*

$$\frac{s}{((2s(r+1))!)^2} \cdot \frac{1}{n^2} \cdot \frac{n!}{(r!)^m}.$$

If m is a fraction of n, in other words, r is a constant and very small compared to the length of the codeword n, then the first term in the formula of Corollary 4, $\frac{s}{((2s(r+1))!)^2}$ is a constant, and the rate of constructed codes are asymptotically optimal. Our constructed codes have size $\Omega(1/n^2)$ as much as the size of $\mathcal{S}_n^{\mathbf{r}}$ for fixed s.

To construct codes of general multipermutations, i.e. where $\mathbf{r} = (r_1, r_2, \ldots, r_m)$. We choose $\mathrm{r}_{\max} = \max_{i \in [1,m]} r_i$ and follow Construction 4 by replacing r by $\mathrm{r}_{\max}$. The following result is immediate.

Corollary 5. *There exists an s-MBSD code in $\mathcal{S}_n^{\mathbf{r}}$ of size at least*

$$\frac{s}{((2s(\mathrm{r}_{\max}+1))!)^2} \cdot \frac{1}{n^2} \cdot \frac{n!}{(r_1!)(r_2!)\cdots(r_m!)}.$$

VI. Permutation Codes Correcting a Burst of Unstable Deletions

This section serves to deal with permutation codes correcting unstable deletions.

A. An Upper Bound

Let $A_{\mathsf{PBUD}}(n, s)$ be the maximum size of an s-PBUD code in $\mathcal{S}_n$. We abbreviate $A_{\mathsf{PBUD}}(n, 1)$ to $A_{\mathsf{PBUD}}(n)$. Gabrys *et al.* [11] gave the following upper bound.

Lemma 1 (Gabrys *et al.* [11]). *For any positive $\epsilon < 1$, there exists an N_ϵ such that for all $n \geq N_\epsilon$, $A_{\mathsf{PBUD}}(n) \leq \frac{n!}{n(n-\log n)}(1+\epsilon)$.*

Gabrys *et al.* [11] constructed a class of 1-PBUD codes of size at least $n!/n^2$, which are asymptotically optimal with respect to this upper bound.

We first present an upper bound for $A_{\mathsf{PBUD}}(n, s)$. Assume that $n = t \cdot s$. Let $\pi = (\pi_1, \pi_2, \ldots, \pi_n) \in \mathcal{S}_n$. A *consecutive run* of π is a substring of maximal length in π that contains consecutively valued symbols, increasing or decreasing. Let $\mathrm{R}(\pi)$ be the number of consecutive runs in π. For example, $\pi = (5, 1, 2, 4, 3, 6)$ has four consecutive runs (5), $(1, 2)$, $(4, 3)$ and (6). It is proved that $|\mathcal{B}_{\downarrow,1}(\pi)| = \mathrm{R}(\pi)$ [11].

Lemma 2. *Let n, s and t be positive integers such that $n = st$. For any permutation $\pi \in \mathcal{S}_n$, we have*

$$\begin{aligned} |\mathcal{B}_{\downarrow,s}(\pi)| &\geq 1 + \sum_{i=1}^{s}(|\mathcal{B}_{\downarrow,1}(\mathsf{p}_i)| - 1) \\ &= 1 + \sum_{i=1}^{s}(\mathrm{R}(\mathsf{p}_i) - 1), \end{aligned}$$

where $\mathsf{p}_i = \mathsf{p}(\pi_{(s,i)})$.

Proof: We prove it by showing a procedure to build the set $\mathcal{B}_{\downarrow,s}(\pi)$.

Recall that $\mathcal{B}_s(\pi)$ is the set of all vectors obtained as a result of a burst of s stable deletions in π. Let $\tilde{\mathcal{B}}_{\downarrow,s}(\pi) = \{\mathsf{p}(\mathbf{u}) : \mathbf{u} \in \mathcal{B}_s(\pi)\}$ be the multiset of all permutations. Here we say multiset since there may exist two different vectors $\mathbf{u}$ giving the same $\mathsf{p}(\mathbf{u})$. Observe that the cardinality $|\mathcal{B}_s(\pi)| = n - s + 1 = |\tilde{\mathcal{B}}_{\downarrow,s}(\pi)|$, and the set $\mathcal{B}_{\downarrow,s}(\pi)$ is just the collection of distinct elements in $\tilde{\mathcal{B}}_{\downarrow,s}(\pi)$. We define the similar multiset for p_i, the permutation projection of the ith row of π when viewed as an $s \times t$ matrix, that is $\tilde{\mathcal{B}}_{\downarrow,1}(\mathsf{p}_i) = \{\mathsf{p}(\mathbf{u}) : \mathbf{u} \in \mathcal{B}_1(\mathsf{p}_i)\}$, and we have $|\mathcal{B}_1(\mathsf{p}_i)| = t = |\tilde{\mathcal{B}}_{\downarrow,1}(\mathsf{p}_i)|$ for all $i \in [1, s]$. Moreover,

$$\sum_{i=1}^{s} |\tilde{\mathcal{B}}_{\downarrow,1}(\mathsf{p}_i)| = t \cdot s = n.$$

Hence,

$$|\tilde{\mathcal{B}}_{\downarrow,s}(\pi)| = n - s + 1 = 1 + \sum_{i=1}^{s}(|\tilde{\mathcal{B}}_{\downarrow,1}(\mathsf{p}_i)| - 1). \quad (1)$$

Let $\mathbf{u}^i = (\pi_1, \ldots, \pi_{i-1}, \pi_{i+s}, \ldots, \pi_n)$, $i \in [1, n-s+1]$. We start with $i = 1$ and consider the element $\mathsf{p}(\mathbf{u}^i)$ in the multi-set $\tilde{\mathcal{B}}_{\downarrow,s}(\pi)$. Let i_0, k_1 be such that $i = i_0 + k_1 \cdot s$. We observe that if there is an index j, where

$i < j \leq n-s+1$ such that $\mathsf{p}(\mathbf{u}^i) = \mathsf{p}(\mathbf{u}^j)$, then the i_0th row of π have $(\mathsf{p}_{i_0})_{\downarrow,\{k_1\}} = (\mathsf{p}_{i_0})_{\downarrow,\{k_2\}}$ for some k_2 satisfying $j \leq i_0 + k_2 \cdot s \leq j+s-1$. In this case, we delete the element $\mathsf{p}(\mathbf{u}^i)$ from the multi-set $\tilde{\mathcal{B}}_{\downarrow,s}(\pi)$ and the element $(\mathsf{p}_{i_0})_{\downarrow,\{k_1\}}$ from the multi-set $\tilde{\mathcal{B}}_{\downarrow,1}(\mathsf{p}_{i_0})$, then increase i by one. Otherwise, we just increase i by one and repeat this process. The procedure ends when $i = n-s+1$.

When the above procedure ends, it returns the set $\mathcal{B}_{\downarrow,s}(\pi)$, which consists of all distinct elements in $\tilde{\mathcal{B}}_{\downarrow,s}(\pi)$. Simultaneously, we also obtain s multi-sets $\hat{\mathcal{B}}_{\downarrow,1}(\mathsf{p}_i) \subset \tilde{\mathcal{B}}_{\downarrow,1}(\mathsf{p}_i)$ for $1 \leq i \leq s$. However, the procedure does not guarantee that $\hat{\mathcal{B}}_{\downarrow,1}(\mathsf{p}_i)$ is a set. We only know $|\mathcal{B}_{\downarrow,1}(\mathsf{p}_i)| \leq |\hat{\mathcal{B}}_{\downarrow,1}(\mathsf{p}_i)|$. Further, each time we delete an element in $\tilde{\mathcal{B}}_{\downarrow,s}(\pi)$, we also delete an element in $\tilde{\mathcal{B}}_{\downarrow,1}(\mathsf{p}_i)$ for some i. Hence, from (1), we have

$$|\mathcal{B}_{\downarrow,s}(\pi)| = 1 + \sum_{i=1}^{s}(|\hat{\mathcal{B}}_{\downarrow,1}(\mathsf{p}_i)| - 1).$$

Since $|\mathcal{B}_{\downarrow,1}(\mathsf{p}_i)| \leq |\hat{\mathcal{B}}_{\downarrow,1}(\mathsf{p}_i)|$, we obtain

$$|\mathcal{B}_{\downarrow,s}(\pi)| \geq 1 + \sum_{i=1}^{s}(|\mathcal{B}_{\downarrow,1}(\mathsf{p}_i)| - 1).$$

Then the proof follows from the fact that $|\mathcal{B}_{\downarrow,1}(\mathsf{p}_i)| = \mathrm{R}(\mathsf{p}_i)$. ■

The following result is useful in the estimate of $A_{\mathsf{PBUD}}(n,s)$.

Lemma 3 (Gabrys *et al.* [11])**.** *The number of permutations in $\mathcal{S}_n$ with at most $n - \log n$ consecutive runs is at most $\frac{n!(n-\log n)^2}{(\log n)!}$.*

We are now ready to provide an upper bound for the maximum size of a permutation code correcting a burst of s unstable deletions.

Theorem 7. *Let s be a fixed positive integer. For any positive $\epsilon < 1$, there exists an N_ϵ such that for all $n \geq N_\epsilon$ and $n = s \cdot t$,*

$$A_{\mathsf{PBUD}}(n,s) \leq (1+\epsilon)\frac{(n-s)!}{s(t-\log t)}.$$

Proof: Suppose that $\mathcal{C} \subseteq \mathcal{S}_n$ is an s-PBUD code. Let $\mathcal{C}_1 = \{\pi \in \mathcal{C} : |\mathcal{B}_{\downarrow,1}(\mathsf{p}_i)| > t - \log t \text{ for all } i \in [1,s]\}$ and $\mathcal{C}_2 = \mathcal{C} \setminus \mathcal{C}_1$. Since $\mathcal{B}_{\downarrow,s}(\pi) \subseteq \mathcal{S}_{n-s}$, by Lemma 2, we have $|\mathcal{C}_1|(s(t-\log t)+1) \leq (n-s)!$. Hence,

$$|\mathcal{C}_1| \leq \frac{(n-s)!}{s(t-\log t)}. \tag{2}$$

Now we estimate the size of $\mathcal{C}_2$. For each $\pi \in \mathcal{C}_2$, there exists at least one i such that $|\mathcal{B}_{\downarrow,1}(\mathsf{p}_i)| \leq t - \log t$. We have s choices for such i, and for each chosen i, there are at most $\frac{t!(t-\log t)^2}{(\log t)!}$ ways to choose p_i by Lemma 3. To simplify the computation, we assume that there is no constraint on the symbols in the remaining $s-1$ rows. Now given p_i in π, let $\tilde{\pi} \in \mathcal{B}_{\downarrow,s}(\pi) \subseteq \mathcal{S}_{n-s}$ be some permutation by suffering a burst of s unstable deletions. Then there are $\binom{n-s}{t-1}$ possibilities of corresponding $t-1$ symbols in $\tilde{\pi}$ (deduced from p_i), and $(n-t-s+1)!$ choices for the remaining $(n-t-s+1)$ symbols in $\tilde{\pi}$. Since $\mathcal{C}_2$ is an s-PBUD code, each $\tilde{\pi}$ is obtained from at most one $\pi \in \mathcal{C}_2$. So we have $\binom{n-s}{t-1}(n-t-s+1)!$ ways to choose the remaining $s-1$ rows of π. Therefore,

$$\begin{aligned}|\mathcal{C}_2| &\leq s \cdot \frac{t!(t-\log t)^2}{(\log t)!} \cdot \binom{n-s}{t-1}(n-t-s+1)! \qquad (3)\\ &\leq \frac{st \cdot (t-\log t)^2 \cdot (n-s)!}{(\log t)!}. \qquad (4)\end{aligned}$$

From (2) and (4), we get

$$\begin{aligned}|\mathcal{C}| &= |\mathcal{C}_1| + |\mathcal{C}_2| \\ &\leq \frac{(n-s)!}{s(t-\log t)} + \frac{st \cdot (t-\log t)^2 \cdot (n-s)!}{(\log t)!} \\ &\leq \frac{(n-s)!}{s(t-\log t)}\Big(1 + \frac{s^2 \cdot t \cdot (t-\log t)^3}{(\log t)!}\Big).\end{aligned}$$

Since $\lim_{t\to\infty}\frac{s^2\cdot t\cdot(t-\log t)^3}{(\log t)!} = 0$ for fixed s, there exists an N_ϵ such that for all $n \geq N_\epsilon$ and $n = s\cdot t$,

$$A_{\mathsf{PBUD}}(n,s) \leq (1+\epsilon)\frac{(n-s)!}{s(t-\log t)}.$$

The theorem is proved. ■

Note that when $s = 1$, the upper bound for $A_{\mathsf{PBUD}}(n,s)$ in Theorem 7 is exactly the same as the upper bound for $A_{\mathsf{PBUD}}(n)$ provided by Gabrys *et al.* [11] (refer to Lemma 1).

B. Code Constructions

In this subsection, we apply the permutation interleaving method to construct s-PBUD codes for $s \geq 2$.

Definition 12. For vectors $\tau^i = (\tau_1^i, \tau_2^i, \ldots, \tau_t^i)$, $i \in [1,s]$, of length t, the *interleaved vector* $\mathbf{u} = \tau^1 \circ \tau^2 \circ \cdots \circ \tau^s$ is obtained by alternatively placing the elements of $\tau^1, \tau^2, \ldots, \tau^s$ in order. That is

$$u_j = \tau^i_{\lceil j/s \rceil}, \quad j \in [1, st]$$

where $i \equiv j \bmod s$. If we view $\mathbf{u}$ as an $s \times t$ array, then τ^i is just the ith row of $\mathbf{u}$. For a class of s codes $\mathcal{C}_i$, $i \in [1,s]$ of same length, the *interleaved code*

$$\mathcal{C}_1 \circ \mathcal{C}_2 \circ \cdots \circ \mathcal{C}_s = \{\tau^1 \circ \tau^2 \circ \cdots \circ \tau^s : \tau^i \in \mathcal{C}_i, i \in [1,s]\}.$$

For any integer a, a vector $\tau = (\tau_1, \tau_2, \ldots, \tau_t)$ and a code $\mathcal{C}$, define $\tau + a = (\tau_1 + a, \tau_2 + a, \ldots, \tau_t + a)$ and $\mathcal{C} + a = \{\tau + a : \tau \in \mathcal{C}\}$.

Since the decoding algorithm for our codes is rather complex for general s, we first consider the case $s = 2$ to explain the idea. To simplify notations, we assume that n is even.

Theorem 8. *Let $t \geq 3$ and $n = 2t$. Suppose that for each $i \in \{1,2\}$, $\mathcal{C}_i \subseteq \mathcal{S}_t$ is a 1-PBUD code. Then the interleaved code $\mathcal{C} = \mathcal{C}_1 \circ (\mathcal{C}_2 + t)$ is a $\leqslant$2-PBUD code in $\mathcal{S}_n$.*

Proof: Suppose we receive a permutation $\tilde{\pi} \in \mathcal{S}_{n-k}$, $1 \leq k \leq 2$. We want to find the unique permutation $\pi = \tau \circ (\upsilon + t) \in \mathcal{C}$ such that $\tilde{\pi}$ is obtained from π through a burst of at most two unstable deletions.

If $k = 1$, that is, $\tilde{\pi} \in \mathcal{S}_{n-1}$, then we know that only one symbol is deleted from π. Let f_1 be the subsequence of $\tilde{\pi}$ with values from $[1, t-1]$, which is a permutation in $\mathcal{S}_{t-1}$. Note that f_1 is obtained from τ by experiencing a single unstable deletion, thus we can recover τ from f_1 since $\mathcal{C}_1$ is a 1-PBUD code. Let f_2 be the subsequence of $\tilde{\pi}$ with values from $[t+1, n-1]$, which is a permutation over $[t+1, n-1]$. Note that values in f_2 are originally from $\upsilon + t$ before the unstable deletion. Thus $f_2 - t$ is a permutation in $\mathcal{S}_{t-1}$ obtained from υ by experiencing a single unstable deletion. Note that $f_2 - t = \tilde{\pi}_{\downarrow, I}$, where $I = \{i \in [1, n-1] : \tilde{\pi}_i \le t\}$. Since $\mathcal{C}_2$ is a 1-PBUD code, we can recover υ from f_2. Hence π is the interleaved vector $\tau \circ (\upsilon + t)$ determined uniquely.

If $k = 2$, that is, the permutation $\tilde{\pi}$ has length $n-2$, then we know that there are exactly two symbols deleted. Since the deletions are adjacent, by definition of $\mathcal{C}$, there is exactly one deleted symbol from τ and $\upsilon + t$. Now we de-interleave permutation $\tilde{\pi}$. Let $f_1 = \tilde{\pi}_{(2,1)}$ and $f_2 = \tilde{\pi}_{(2,2)}$. Then it must be the case that $f_1 \in \mathcal{S}_{t-1}$ and f_2 is a permutation over $[t, n-2]$. Similarly, we can recover τ and υ from f_1 and $f_2 - (t-1)$ respectively, where $f_2 - (t-1)$ is actually $\tilde{\pi}_{\downarrow, I}$ with $I = \{i \in [1, n-2] : \tilde{\pi}_i \le t-1\}$. Thus $\pi = \tau \circ (\upsilon + t)$ is uniquely determined.

Therefore, the interleaved code $\mathcal{C} = \mathcal{C}_1 \circ (\mathcal{C}_2 + t)$ is a 2-PBUD code in $\mathcal{S}_n$. ∎

We are now ready to give the decoding algorithm for the codes in Theorem 8.

Algorithm 1 $\mathcal{S}_{n-k} \to \mathcal{S}_n, 2 \ge k \ge 1$

```
Require: $t \ge 3, n = 2t, \tilde{\pi} = (\tilde{\pi}_1, \tilde{\pi}_2 \dots, \tilde{\pi}_{n-k})$
Ensure: $\pi = (\pi_1, \pi_2, \dots, \pi_n) \in \mathcal{C}$
    $t' \longleftarrow n - k - t + 1$
    $f_1 \longleftarrow$ subsequence of $\tilde{\pi}$ with entries from $[1, t-1]$
    $\tau \longleftarrow$ correcting $f_1$ by a decoder of $\mathcal{C}_1$
    $f_2 \longleftarrow$ subsequence of $\tilde{\pi}$ with entries from $[t'+1, n-k]$
    $\upsilon \longleftarrow$ correcting $f_2 - t'$ by a decoder of $\mathcal{C}_2$
    $\pi \longleftarrow \tau \circ (\upsilon + t)$
```

We illustrate the preceding algorithm with the following example.

Example 9. Let $t = 4$ and $n = 8$. Suppose that $\tau = (2, 1, 4, 3) \in \mathcal{C}_1$ and $\upsilon = (1, 3, 4, 2) \in \mathcal{C}_2$. Then $\pi = \tau \circ (\upsilon + t) = (2, 5, 1, 7, 4, 8, 3, 6) \in \mathcal{C}$.

If only one symbol 4 is deleted from π, then $\tilde{\pi} = (2, 4, 1, 6, 7, 3, 5)$ is received. By Algorithm 1, $t' = 4$, $f_1 = (2, 1, 3)$, and $f_2 = (6, 7, 5)$. From f_1 and $f_2 - 4 = (2, 3, 1)$, we can recover $\tau = (2, 1, 4, 3)$ and $\upsilon = (1, 3, 4, 2)$ by the decoders of $\mathcal{C}_1$ and $\mathcal{C}_2$ in [11], respectively. Hence $\pi = \tau \circ (\upsilon + t)$ is determined in this case.

If two adjacent symbols 7 and 4 are deleted, then $\tilde{\pi} = (2, 4, 1, 6, 3, 5)$ is received. By Algorithm 1, $t' = 3$, $f_1 = (2, 1, 3)$, and $f_2 = (4, 6, 5)$. From f_1 and $f_2 - 3 = (1, 3, 2)$, again we can recover $\tau = (2, 1, 4, 3)$ and $\upsilon = (1, 3, 4, 2)$ by the decoders of $\mathcal{C}_1$ and $\mathcal{C}_2$. Hence $\pi = \tau \circ (\upsilon + t)$ is also uniquely determined in this case.

Now we generalize the interleaving method in Theorem 8 to construct s-PBUD codes. We sketch the main idea of the decoding algorithm first. In Theorem 9, the code is constructed by interleaving s 1-PBUD codes. Suppose there is a burst of k unstable deletions in codeword $\mathbf{c} \in \mathcal{S}_n$, and we receive a permutation $\pi \in \mathcal{S}_{n-k}$ with $k < s$. The main task is to find an appropriate permutation $\pi' \in \mathcal{S}_{n-s}$ which is obtained from π by experiencing a burst of $s-k$ unstable deletions, and in such a way that π' is simultaneously obtained from the original permutation $\mathbf{c}$ by suffering a burst of s unstable deletions. Hence we can recover $\mathbf{c}$ by correcting the de-interleaved components of π'. Note that values from $[1, t-1]$ and $[n-k-t+2, n-k]$ in π are originally from $[1, t]$ and $[n-t+1, n]$ in $\mathbf{c}$, which are periodically placed in $\mathbf{c}$, respectively. The permutation π' is obtained by carefully checking the change of placement of these values in π.

Theorem 9. *Let $t, s \ge 3$ and $n = st$. For each $i \in [1, s]$, let $\mathcal{C}_i$ be a 1-PBUD code over $[1, t]$. Then the interleaved code $\mathcal{C} = \mathcal{C}_1 \circ (\mathcal{C}_2 + t) \circ \cdots \circ (\mathcal{C}_i + (i-1)t) \circ \cdots \circ (\mathcal{C}_s + (s-1)t)$ is an $\leqslant$s-PBUD code over $[1, n]$.*

Proof: Suppose that a codeword $\mathbf{c} = \tau^1 \circ (\tau^2 + t) \circ \cdots \circ (\tau^i + (i-1)t) \circ \cdots \circ (\tau^s + (s-1)t) \in \mathcal{C}$, where $\tau^i \in \mathcal{C}_i$ for $i \in [1, s]$, suffers a burst of unstable deletions and a permutation $\pi = (\pi_1, \dots, \pi_{n-k}) \in \mathcal{S}_{n-k}$ is received. We show that $\mathbf{c}$ is uniquely identifiable from π as follows. Let $P_i = [(i-1)t + 1, it]$ for each $i \in [1, s]$.

Case 1 ($k = s$): In this case, exactly one symbol in each P_i is deleted from $\mathbf{c}$ since the unstable deletions are adjacent. By de-interleaving the permutation π, we have $f_i = \pi_{(s,i)}$ for each $i \in [1, s]$. Then τ^i is uniquely determined from the permutation $f_i - (i-1)(t-1)$ by a decoder of $\mathcal{C}_i$ and consequently $\mathbf{c}$ is recovered. Note that $f_i - (i-1)(t-1)$ is actually $\mathsf{p}(\pi_{(s,i)})$.

Case 2 ($k < s$): In this case, we will get $\pi' = \pi_{\downarrow, I}$ for some positions set I of size $s-k$ and then apply Case 1 with π' to recover $\mathbf{c}$. Since there is a burst of k $(< s)$ unstable deletions from $\mathbf{c}$, there is at most one symbol in each P_i that is deleted from $\mathbf{c}$. We consider the positions of symbols from $[1, t-1]$ in π, since they are originally from $[1, t]$ in $\mathbf{c}$. Now collect these $(t-1)$ positions of symbols from $[1, t-1]$ in π, and denote them by k_i, $i \in [1, t-1]$ such that $k_i < k_{i+1}$. Note that in the original permutation $\mathbf{c}$, each $k_i = (i-1)s + 1$, $i \in [1, t-1]$ and $k_{i+1} - k_i = s$. But in π, they maybe not anymore due to the deletion errors. Define $k_t := n - k + 1$ and let $d_j = k_{j+1} - k_j$, $j \in [1, t-1]$. We will roughly determine the place of the k unstable deletions happened in $\mathbf{c}$ based on the first abnormal value k_i and the unique abnormal value d_j. Note that $k_1 \le s + 1$.

Suppose $1 < k_1 \le s$. Then the deleted locations are among the positions $[1, s]$ in $\mathbf{c}$. Let $\pi' = \pi_{\downarrow, [1, s-k]}$.

Suppose $k_1 = s + 1$. Then there must be a unique $j \in [1, t-1]$ such that $d_j = s - k$, and for all $i \in [1, t-1] \setminus \{j\}$, $d_i = s$. Let $\pi' = \pi_{\downarrow, [k_j, k_{j+1} - 1]}$.

Suppose $k_1 = 1$. There are only two possible cases: (1) there is a unique $j \in [1, t-1]$ such that $d_j = s - k$; (2) there is a unique $j \in [1, t-1]$ such that $d_j = 2s - k$. If (1) happens, let $\pi' = \pi_{\downarrow, [k_j, k_{j+1} - 1]}$. If (2) happens, let $R = [k_j, k_{j+1} - 1]$. Note that $|R| = 2s - k$ and a burst of unstable deletions have occurred in some positions from R in π. Now we need to check the positions h_i of values from $[n - k - t + 2, n - k]$ in π such that $h_i < h_{i+1}$,

$i \in [1, t-1]$. By similarly considering differences between h_i and h_{i+1}, we can recover $\mathbf{c}$ for almost all cases except one case where a set $R' = [h_j, h_{j+1} - 1]$ of $2s - k$ positions in π is found for correction. Let $D = R \cap R'$, then errors occur in D. Suppose $|D| < s$, then let $\pi' = \pi_{\downarrow,[k_j,h_{j+1}]}$ if $h_j < k_j$ and $\pi' = \pi_{\downarrow,[h_j+1,k_{j+1}-1]}$ if $h_j > k_j$. It comes to the worst case if $|D| > s$. In this case, $h_j = k_j - 1$ and $h_{j+1} = k_{j+1} - 1$. We split it in two cases further.

Case 2a ($k = 1$): If $\pi_{k_{j+1}-s} \in P_1$, then we know that an error has occurred in a position from $[k_j, h_{j+1} - s]$ in π. Let $\pi' = \pi_{\downarrow,[k_j,h_{j+1}-s]}$. If not, let $\pi' = \pi_{\downarrow,[k_j+s,h_{j+1}]}$.

Case 2b ($1 < k < s$): We know that $\pi_{k_j} \in P_1$. If for all $i \in [1, s-1]$, $\pi_{k_j+i} \in P_i \cup P_{i+1}$, then let $\pi' = \pi_{\downarrow,[k_j+s,h_{j+1}]}$. Otherwise, find the smallest index $i \in [1, s-1]$ such that $\pi_{k_j+i} \notin (P_i \cup P_{i+1})$. Let $\pi' = \pi_{\downarrow,[k_j+i,k_j+i+s-k-1]}$.

Therefore, the interleaved code $\mathcal{C}$ is an $\leqslant s$-PBUD code over $[1, n]$. ■

We only present an algorithm of computing π' from π when $k < s$ as Case 2 in the proof of Theorem 9. Once π' is computed, it is easy to recover the codeword $\mathbf{c}$ by applying Case 1.

Algorithm 2 $\mathcal{S}_{n-k} \to \mathcal{S}_{n-s}, s > k \geq 1$

```
Require: $t \geq 3, n = st, \pi = (\pi_1, \pi_2 \ldots, \pi_{n-k})$
Ensure: $\pi' = (\pi'_1, \pi'_2, \ldots, \pi'_{n-s})$
  $t' \longleftarrow n - k - t + 2$
  $k_j \longleftarrow$ positions of entries $[1, t-1]$ in $\pi$, $j \in [1, t-1]$
  $h_j \longleftarrow$ positions of entries $[t', n-k]$ in $\pi$, $j \in [1, t-1]$
  $k_t \longleftarrow n - k + 1$; $k_0 \longleftarrow 1$
  for $j \in [0, t-1]$ do
    $d_j \longleftarrow k_{j+1} - k_j$
    if $d_j = s - k$ then
      return $\pi' \longleftarrow \pi_{\downarrow,[k_j,k_{j+1}-1]}$
    else if $d_j = 2s - k$ then
      $a \longleftarrow k_j, b \longleftarrow k_{j+1}$
  for $j \in [1, t-1]$ do
    if $a < h_j < a + s - 1$ then
      return $\pi' \longleftarrow \pi_{\downarrow,[a,h_j]}$
    else if $h_j = a + s - 1$ then
      return $\pi' \longleftarrow \pi_{\downarrow,[h_j+1,b-1]}$
  if $k = 1$ then
    if $\pi_{b-s} \leq t$ then
      return $\pi' \longleftarrow \pi_{\downarrow,[a,b-s-1]}$
    else
      return $\pi' \longleftarrow \pi_{\downarrow,[a+s,b-1]}$
  if $k > 1$ then
    for $i \in [1, s-1]$ do
      if $\pi_{a+i} \notin [(i-1)t + 1, (i+1)t]$ then
        return $\pi' \longleftarrow \pi_{\downarrow,[a+i,a+i+s-k-1]}$
    return $\pi' \longleftarrow \pi_{\downarrow,[a+s,b-1]}$
```

Now we give an example to illustrate Case 2b, which is the worst case in the proof of Theorem 9.

Example 10. Let $t = 5$, $s = 3$ and $n = 15$. Then $P_1 = [1, 5]$, $P_2 = [6, 10]$ and $P_3 = [11, 15]$. Suppose that $\tau^1 = (3, 1, 2, 5, 4) \in \mathcal{C}_1$, $\tau^2 = (2, 5, 1, 4, 3) \in \mathcal{C}_2$ and $\tau^3 = (4, 2, 5, 1, 3) \in \mathcal{C}_3$. Then $\mathbf{c} = (3, 7, 14, 1, 10, 12, 2, 6, 15, 5, 9, 11, 4, 8, 13) \in \mathcal{C}$.

Let $k = 2$. Suppose the two adjacent symbols 12 and 2 are deleted, then $\pi = (2, 6, 12, 1, 9, 5, 13, 4, 8, 10, 3, 7, 11)$ is received. By checking the positions of values from $[1, 4]$ and $[10, 13]$ in π, we have $R = [4, 7]$ and $R' = [3, 6]$. That is $a = k_j = 4$, $b = 8$ in Algorithm 2, and there is no h_j in $[a, a + s - 1] = [4, 6]$. Since the smallest index $i \in [1, 2]$ such that $\pi_{k_j+i} \notin (P_i \cup P_{i+1})$ is 2, $\pi' = (2, 5, 11, 1, 8, 12, 4, 7, 9, 3, 6, 10) = \pi_{\downarrow,\{6\}}$. De-interleaving π', we have $f_1 = (2, 1, 4, 3)$, $f_2 = (5, 8, 7, 6)$ and $f_3 = (11, 12, 9, 10)$. Thus τ^i, $i = 1, 2, 3$ can be recovered from f_1, $f_2 - 4 = (1, 4, 3, 2)$ and $f_3 - 8 = (3, 4, 1, 2)$ by decoders of $\mathcal{C}_i$ respectively, and consequently $\mathbf{c}$ is uniquely determined.

Remark 3. The code we construct in Theorem 9 has size at least $\left(\frac{(n/s)!}{(n/s)^2}\right)^s$. Although this is not optimal with respect to the upper bound we derive in Theorem 7, its rate is asymptotically

$$\frac{s \ln (n/s)! - 2s \ln (n/s)}{\ln n!} \sim \frac{(n-2s) \ln n + O(n)}{n \ln n + O(n)} \sim 1,$$

for fixed s.

VII. Multipermutation Codes Correcting a Burst of Unstable Deletions

In this section, we present our method to construct multipermutation codes over $\mathcal{S}_n^{\mathbf{r}}$ that can correct a burst of s unstable deletions. We first consider the case of regular multipermutations and $s \leq r$.

Sala *et al.* [12] provided a method to combat single unstable deletion in multipermutations by using interleaved codes and VT codes. They also provided a method to combat multiple deletions. However, their method is based on multipermutation codes of certain Hamming distance. To construct a code that is capable of correcting at most s deletions, they use interleaved codes over $(s+1)(2s+1) + 1 = \Theta(s^2)$ different congruent classes. In this section, we give a direct construction, using only three different congruent classes (mod 3). We can construct multipermutation codes that are capable of correcting a burst of s unstable deletions. Our method is to convert unstable deletions to stable deletions and then use the decoding algorithm for s-MBSD code to recover the original codeword.

Recall that the signature of $\mathbf{u} = (u_1, u_2, \ldots, u_n)$ is the binary vector $\alpha(\mathbf{u}) = (\alpha(u_1), \ldots, \alpha(u_{n-1}))$ of length $n - 1$, where $\alpha(u_i) = 1$ if $u_{i+1} \geq u_i$, and 0 otherwise, for all $i \in [1, n-1]$.

Construction 5. *Given $n = rm = (r+1)st$. Assume that $3|(r+1)t$, t is even and $s \leq r$. Let $a \in \mathbb{Z}_{(r+1)t}, b \in \mathbb{Z}_m$ and $c, d \in \mathbb{Z}_{(2s(r+1))!}$. Denote $k = (r+1)t/3$. Let $\mathcal{D}_s(n; r, a, b, c, d)$ be the set of all regular multipermutations $\sigma \in \mathcal{S}_n^{\mathbf{r}}$ such that the following holds:*

(i) **Symbol constraint.**

(1) $\sigma_i \equiv 0 \pmod 3$ *for* $i \in [1, n/3]$,

(2) $\sigma_i \equiv 1 \pmod 3$ *for* $i \in [n/3 + 1, 2n/3]$,

(3) $\sigma_i \equiv 2 \pmod 3$ *for* $i \in [2n/3 + 1, n]$,

(4) $\sigma_{(k-1)s+1} \neq \sigma_{ks+1}$ *and* $\sigma_{(2k-1)s+1} \neq \sigma_{2ks+1}$.

(ii) **The first row constraint.** *When we view σ as an $s \times (r+1)t$ array, the first row $\sigma_{(s,1)}$ and its signature, $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_{(r+1)t-1})$, satisfy*

(1) $\sigma_{(s,1)} \in \mathcal{C}_{a,b}((r+1)t; m)$,

(2) $(\alpha_{k-1}, \alpha_k, \alpha_{k+1}) \in \{(1,0,1),(0,1,0)\}$, *and* $(\alpha_{2k-1}, \alpha_{2k}, \alpha_{2k+1}) \in \{(1,0,1),(0,1,0)\}$.

(iii) **The adjacent columns constraint.** *When we view σ as an $s(r+1) \times t$ array, the pairs of adjacent columns satisfy*

(1) $\sum_{i=1}^{t/2} \mu(\sigma^{(s(r+1),2i-1)} || \sigma^{(s(r+1),2i)}) \equiv c \pmod{(2s(r+1))!}$,

(2) $\sum_{i=1}^{t/2-1} \mu(\sigma^{(s(r+1),2i)} || \sigma^{(s(r+1),2i+1)}) \equiv d \pmod{(2s(r+1))!}$.

Theorem 10. *The multipermutation code $\mathcal{D}_s(n; r, a, b, c, d)$ from Construction 5 is a regular s-MBUD code over $\mathcal{S}_n^{\mathbf{r}}$.*

Proof: We establish the theorem by giving an efficient decoding algorithm for $\mathcal{D}_s(n; r, a, b, c, d)$ that recovers from a burst of exact s unstable deletions.

Suppose a burst of s unstable deletions occurs in a codeword $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)$ in $\mathcal{D}_s(n; r, a, b, c, d)$, giving $\tilde{\sigma} = (\tilde{\sigma}_1, \tilde{\sigma}_2, \ldots, \tilde{\sigma}_{n-s})$. Further suppose a bust of s stable deletions occurs in the same positions of σ, giving $\sigma' = (\sigma'_1, \sigma'_2, \ldots, \sigma'_{n-s})$. Since $s \leq r$, $\tilde{\sigma}$ can be obtained by increasing or decreasing the values of each symbol of σ' at most one, and vice versa. Our strategy is to recover σ' from $\tilde{\sigma}$ and then apply the decoding algorithm for stable deletions (described in the proof of Theorem 6) on σ' to recover the codeword σ.

We first view σ as an $s \times 3k$ array, and σ' as an $s \times (3k-1)$ array. Since the symbols in the first $k-1$ columns of σ' come from the first ks symbols of σ, they belong to the first congruent class, where the values are congruent to 0 (mod 3). We then recover these symbols as follows. For $i \in [1, (k-1)s]$,

$$\sigma'_i = \begin{cases} \tilde{\sigma}_i, & \text{if } \tilde{\sigma}_i \equiv 0 \pmod 3; \\ \tilde{\sigma}_i - 1, & \text{if } \tilde{\sigma}_i \equiv 1 \pmod 3; \\ \tilde{\sigma}_i + 1, & \text{if } \tilde{\sigma}_i \equiv 2 \pmod 3. \end{cases}$$

Similarly, for $i \in [ks+1, (2k-1)s] \cup [2ks+1, (3k-1)s]$, the symbols σ'_i can be recovered in the same way.

Now we need to recover the symbols σ'_i for $i \in [(k-1)s+1, ks] \cup [(2k-1)s+1, 2ks]$. Since $\sigma_{(s,1)} \in \mathcal{C}_{a,b}((r+1)t; m)$, using the decoding algorithm of VT codes, we are able to find the run of α in which the deletion occurs. Suppose that this run is $(\alpha_{j_0}, \alpha_{j_0+1}, \ldots, \alpha_{j_0+\ell-1})$. According to the condition $(ii.2)$ of Construction 5, we proceed in the following cases.

Case 1: $j_0 + \ell - 1 \leq k - 1$. Then the s deletions occur in the first k columns of σ. It follows that for each $i \in [(k-1)s+1, ks]$ the symbol σ'_i belongs to the second congruent class, and for each $i \in [(2k-1)s+1, 2ks]$ the symbol σ'_i belongs to the third congruent class. Hence we can recover the vector σ' from $\tilde{\sigma}$ and then recover σ.

Case 2: $j_0 = k$ and $\ell = 1$. In this case the deletion in the first row of σ is either $\sigma_{(k-1)s+1}$ or σ_{ks+1}. Since $\sigma_{(k-1)s+1} \neq \sigma_{ks+1}$, we can distinguish this two subcases.

If $\sigma_{(k-1)s+1}$ is deleted, then all the s deletions occur in the first k columns of σ and we can recover σ the same way as in Case 1.

If σ_{ks+1} is deleted, then the s deletions occur in the kth and $(k+1)$th columns of σ. So for each $i \in [(2k-1)s+1, 2ks]$ the symbol σ'_i belongs to the third congruent class and we can recover these symbols. Now in the $s \times rt$ array of σ, we actually already recover all except the kth and $(k+1)$th columns of σ. These two columns span at most two adjacent columns in the $rs \times t$ array σ and thus we can use the condition (iii) to recover these two columns.

Case 3: $k+1 \leq j_0$ and $j_0 + \ell - 1 \leq 2k - 1$. Then the s deletions occur in the second k columns of σ. It follows that for each $i \in [(k-1)s+1, ks]$ the symbol σ'_i belongs to the first congruent class, and for each $i \in [(2k-1)s+1, 2ks]$ the symbol σ'_i belongs to the third congruent class. Hence we can recover σ' and then σ.

Case 4: $j_0 = 2k$ and $\ell = 1$. Then the deletion in the first row of σ is either $\sigma_{(2k-1)s+1}$ or σ_{2ks+1}. We can proceed similarly as in Case 2.

Case 5: $j_0 \geq 2k+1$. This case is similar to Case 1 and Case 3. We know that the s deletions occur in the last k columns of σ and then we can recover σ' and then σ. ■

Remark 4. Let us comment on the cardinality of the constructed codes $\mathcal{D}_s(n; r, a, b, c, d)$. The parameters a, b, c, d form $(r+1)tm((2s(r+1))!)^2$ cosets among the multipermutations in $\mathcal{S}_n^{\mathbf{r}}$. On the other hand, the symbol constraints (i)-(2) and (ii)-(2) cost only a constant redundancy. By placing three groups of $n/3$ multipermutation elements in different parts of each multipermutation, we have $(n/3)!^3/(r!)^m$ multipermutations to choose from. Therefore, by the pigeonhole principle, there exists at least one code with cardinality

$$\begin{aligned} |\mathcal{D}_s(n; r, a, b, c)| &\geq \frac{1}{(r+1)tm(2s(r+1)!)^2} \cdot \frac{\left((\frac{n}{3})!\right)^3}{(r!)^m} \cdot O(1) \\ &= \frac{sr}{n^2.(2s(r+1)!)^2} \cdot \frac{\left((\frac{n}{3})!\right)^3}{(r!)^m} \cdot O(1). \end{aligned}$$

For simplicity, we remove the constant factor at the end of the formula and use notation $\gtrsim$ to represent the lower bound.

Corollary 6. *There exist a, b, and c such that*

$$|\mathcal{D}_s(n; r, a, b, c)| \gtrsim \frac{sr}{n((2s(r+1)!)^2} \cdot \frac{\left((\frac{n}{3})!\right)^3}{(r!)^m}.$$

To construct codes for general multipermutations, i.e. where $\mathbf{r} = (r_1, r_2, \ldots, r_m)$. Let $\mathrm{r}_{\max} = \max_{i \in [1,m]} r_i$ and $\mathrm{r}_{\min} = \min_{i \in [1,m]} r_i$. We follow Construction 5. Our codes can correct a burst of s unstable deletions where $s \leq \mathrm{r}_{\min}$. In the third constraint of the new construction, we view σ as an $s\mathrm{r}_{\max} \times t$ instead. The following result is immediate.

Corollary 7. *There exists an s-MBUD code of size M where*

$$M \gtrsim \frac{s}{n((2s(\mathrm{r}_{\max}+1))!)^2} \cdot \frac{((n/3)!)^3}{(r_1!)(r_2!)\cdots(r_m!)}.$$

We can generalize Construction 5 to the case of $s > r$. Previously, when $s \leq r$, a burst of s unstable deletions increases or decreases the value of those remaining symbols by at most 1. In general, a burst of s unstable deletions increases or decreases the value of those remaining symbols by at most $\lceil s/r \rceil$. We then divide the set of symbols into different congruent classes modulo $2\lceil s/r \rceil + 1$ and construct s-MBUD codes similarly as described in Construction 5.

VIII. Conclusion

In this paper, we investigate permutation and multipermutation codes that are capable of correcting a burst of deletions. For stable deletion, we construct a family of permutation codes of size $\Omega(n!/n^2)$ and a family of multipermutation codes of size $\Omega\Big(n!/\big((r_1!)(r_2!)\dots(r_m!)n^2\big)\Big)$, which have bigger size than previously known. For unstable deletion, in particular, for multipermutations, we extend the method in [12] to construct the first family of codes capable of correcting a burst of unstable deletions. For all constructions, our decoding algorithms have linear running time.

Acknowledgement

The authors would like to thank the editor and the reviewers for their constructive feedback and helpful suggestions.

References

[1] D. Slepian, "Permutation modulation," *Proc. IEEE*, vol. 53, no. 3, pp. 228–236, Mar. 1965.
[2] A. J. H. Vinck, "Coded modulation for power line communications," *AEU Int. J. Electron. Commun.*, vol. 54, no. 1, pp. 45–49, Apr. 2011.
[3] 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.
[4] A. Jiang, M. Schwartz, and J. Bruck, "Correcting charge-constrained errors in the rank-modulation scheme," *IEEE Trans. Inf. Theory*, vol. 56, no. 5, pp. 2112–2120, May 2010.
[5] A. Barg and A. Mazumdar, "Codes in permutations and error correction for rank modulation," *IEEE Trans. Inf. Theory*, vol. 56, no. 7, pp. 3158–3165, Jul. 2010.
[6] E. Gad, M. Langberg, M. Schwartz, and J. Bruck, "Constant-weight Gray Codes for local rank modulation," *IEEE Trans. Inf. Theory*, vol. 57, no. 11, pp. 7431–7442, Nov. 2011.
[7] F. Farnoud, V. Skachek, and O. Milenkovic, "Errorcorrection in flash memories via codes in the Ulam metric," *IEEE Trans. Inf. Theory*, vol. 59, no. 5, pp. 3003–3020, May 2013.
[8] H. Zhou, M. Schwartz, A. A. Jiang, and J. Bruck, "Systematic error-correcting codes for rank modulation," *IEEE Trans. Inf. Theory*, vol. 61, no. 1, pp. 17–32, Jan. 2015.
[9] E. E. Gad, A. Jiang, and J. Bruck, "Trade-offs between instantaneous and total capacity in multi-cell flash memories," in *Proc. IEEE Int. Symp. Inf. Theory*, Cambridge, MA, USA, Jul. 2012, pp. 990–994.
[10] V. I. Levenshtein, "On perfect codes in deletion and insertion metric," *Discrete Math. Appl.*, vol. 2, no. 3, pp. 241–258, 1992.
[11] R. Gabrys, E. Yaakobi, F. Farnoud, F. Sala, J. Bruck, and L. Dolecek, "Codes correcting erasures and deletions for rank modulation," *IEEE Trans. Inf. Theory*, vol. 62, no. 1, pp. 136–150, Jan. 2016.
[12] F. Sala, R. Gabrys, and L. Dolecek, "Deletions in multipermutations," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun. 2014, pp. 2769–2773.
[13] J.-D. Lee, S.-H. Hur, and J.-D. Choi, "Effects of floating-gate interference on NAND flash memory cell operation," *IEEE Electron Device Lett.*, vol. 23, no. 5, pp. 264–266, May 2002.
[14] K. Prall, "Scaling non-volatile memory below 30 nm," in *Proc. 22nd IEEE Non-Volatile Semiconductor Memory Workshop*, Aug. 2007, pp. 5–10.
[15] H. Han, J. Mu, X. Jiao, and Y.-C. He, "Codes correcting a burst of deletions for permutations and multi-permutations," *IEEE Commun. Lett.*, vol. 22, no. 10, pp. 1968–1971, Oct. 2018.
[16] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet Phys. Doklady*, vol. 10, no. 10, pp. 707–710, Feb. 1966.
[17] G. Tenengolts, "Nonbinary codes, correcting single deletion or insertion," *IEEE Trans. Inf. Theory*, vol. 30, no. 5, pp. 766–769, Sep. 1984.
[18] C. Schoeny, A. Wachter-Zeh, R. Gabrys, and E. Yaakobi, "Codes correcting a burst of deletions or insertions," *IEEE Trans. Inf. Theory*, vol. 63, no. 4, pp. 1971–1985, Apr. 2017.
[19] Y. M. Chee, V. K. Vu, and X. Zhang, "Permutation codes correcting a single burst deletion I: Unstable deletions," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun. 2015, pp. 1741–1745.
[20] Y. M. Chee, S. Ling, T. T. Nguyen, V. K. Vu, and H. Wei, "Permutation codes correcting a single burst deletion II: Stable deletions," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun. 2017, pp. 2688–2692.

Yeow Meng Chee (SM'08) received the B.Math. degree in computer science and combinatorics and optimization and the M.Math. and Ph.D. degrees in computer science from the University of Waterloo, Waterloo, ON, Canada, in 1988, 1989, and 1996, respectively. Currently, he is a Professor in the Department of Industrial Systems Engineering and Management, and Associate Vice President of Innovation and Enterprise at the National University of Singapore. Prior to this, he was a Professor in the School of Physical and Mathematical Sciences, and Interim Dean of Science at the Nanyang Technological University. His research interest lies in the interplay between combinatorics and computer science/engineering, particularly combinatorial design theory, coding theory, extremal set systems, and electronic design automation.

San Ling received the B.A. degree in mathematics from the University of Cambridge and the Ph.D. degree in mathematics from the University of California, Berkeley. Since April 2005, he has been a Professor with the Division of Mathematical Sciences, School of Physical and Mathematical Sciences, in the Nanyang Technological University (NTU), Singapore, where he currently holds the President's Chair in Mathematical Sciences. Prior to NTU Singapore, he was with the Department of Mathematics, National University of Singapore. His research interest includes the application of number theory to combinatorial designs, coding theory, cryptography and sequences.

Tuan Thanh Nguyen received the B.Sc. degree and the Ph.D. degree in mathematics from Nanyang Technological University, Singapore, in 2014 and 2019, respectively. Currently, he is a Research Fellow at School of Physical and Mathematical Sciences at Nanyang Technological University, Singapore. His research interests include combinatorics, coding theory, and codes for DNA-based data storage.

Van Khu Vu received his B.Sc. degree in mathematics from Vietnam National University (VNU), Hanoi, in 2010 and the Ph.D. degree in mathematics from Nanyang Technological University (NTU), Singapore in 2018. From 2010 to 2012, he was a lecturer at VNU College of Sciences, Hanoi. Currently, he is a Research Fellow at School of Physical and Mathematical Sciences, NTU, Singapore. His primary research interests lies in the areas of algorithms, combinatorics and coding theory.

Hengjia Wei received the Ph.D. degree in Applied Mathematics from Zhejiang University, Hangzhou, Zhejiang, P. R. China, in 2014. He was a postdoctoral fellow in the Capital Normal University, Beijing, P. R. China, from 2014 to 2016, and a research fellow in the School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore, from 2016 to 2019. Currently, he is a postdoctoral fellow in the Department of Electrical and Computer Engineering, Ben-Gurion University of the Negev, Israel. His research interests include combinatorial design theory, coding theory and their intersections.

Dr. Wei received the 2017 Kirkman Medal from the Institute of Combinatorics and its Applications.

Xiande Zhang received the Ph.D. degree in mathematics from Zhejiang University, Hangzhou, Zhejiang, P. R. China in 2009. From 2009 to 2015, she held postdoctoral positions in Nanyang Technological University and Monash University. Currently, she is a Research Professor at school of Mathematical Sciences, University of Science and Technology of China. Her research interests include combinatorial design theory, coding theory, cryptography, and their interactions. She received the 2012 Kirkman Medal from the Institute of Combinatorics and its Applications. She is now on the editorial board of Journal of Combinatorial Designs.