

Improved constructions of permutation and multi-permutation codes correcting a burst of stable deletions

Yubo Sun, Yiwei Zhang, and Gennian Ge

Abstract—Permutation codes and multi-permutation codes have been widely considered due to their various applications, especially in flash memory. In this paper, we consider permutation codes and multi-permutation codes against a burst of stable deletions. In particular, we propose a construction of permutation codes correcting a burst stable deletion of length s , with redundancy $\log n + 2 \log \log n + O(1)$. Compared to the previous known results, our improvement relies on a different strategy to retrieve the missing symbol on the first row of the array representation of a permutation. We also generalize our constructions for multi-permutations and the variable length burst model. Furthermore, we propose a linear-time encoder with optimal redundancy for single stable deletion correcting permutation codes.

Index Terms—Permutation codes, multi-permutation codes, q -ary codes, flash memory, burst deletion.

I. INTRODUCTION

Flash memory is now widely used, especially in portable storage devices, due to its physical reliability, high storage density, and relatively low cost. Coding schemes for error-correction in flash memory have received considerable attention in recent years. In particular, the *rank modulation scheme*, which uses permutations to encode messages, has been proposed by Jiang et al. [23] to cope with errors caused by charge leakage or charge overshooting during programming. Flash memories are comprised of blocks of cells, and in the rank modulation scheme information is characterized by the relative charge order among a group of cells instead of their absolute charge levels. In this setup, each group of cells induces a permutation. Therefore, the research on permutation codes under different metrics is one of the main topics in the rank modulation scheme of flash memories [3], [10], [11], [22], [24], [47]. Afterwards, multi-permutation codes,

as a generalization of permutation codes with more flexibility, are also introduced in the rank modulation scheme of flash memories [11], [16], [35], [36].

When reading and comparing the charge levels of the cells, some cells might be corrupted and the relative order of charge levels cannot be read correctly. This leads to either erasures or deletions in the permutation, depending on whether the erroneous coordinates are known or unknown. Gabrys et al. [12] formally proposed the models of stable/unstable erasures/deletions in permutation codes for flash memories. Here the term *stable*, also known as *symbol-invariant*, refers to the case that after deletion/erasure the remaining symbols stay invariant. The term *unstable*, also known as *permutation-invariant*, refers to the case that after deletion/erasure the remaining symbols are sorted again, i.e., a new permutation is formed based on the relative orders among the remaining cells. For example, consider a permutation $(2, 3, 1, 5, 4)$ and assume one erasure/deletion error occurs on the second coordinate. A stable erasure will result in $(2, ?, 1, 5, 4)$ and an unstable erasure will result in $(2, ?, 1, 4, 3)$. Similarly, a stable deletion will result in $(2, 1, 5, 4)$ while an unstable deletion will result in $(2, 1, 4, 3)$. See later for a formal definition. In particular, the unstable erasure model is a quite common event in flash memory and it was shown to be equivalent to the stable deletion model [12]. As the stable deletion model is simpler in its mathematical formulations, more subsequent works were focused on the stable deletion model rather than the unstable erasure model. Afterwards, these concepts were further extended to multi-permutation codes by Sala et al. [36].

For single deletion in permutation codes, Levenshtein [28] showed that a set of permutations whose signatures form a binary VT code is capable of correcting a single stable deletion. Gabrys et al. [12] also used binary VT codes to design asymptotically optimal single unstable deletion correcting permutation codes. For multiple deletions in permutation codes, researchers focus on burst deletions. One of the most important reasons is that deletions tend to occur in consecutive cells in flash memories due to capacitive coupling [25], [34]. In fact, not only in flash memories, but also in DNA storage [13], [37] and racetrack memories [8], errors manifest themselves in burst and tend to cluster together. Note that the term of burst error is slightly different between substitution error models and deletion error models. To avoid confusion, we clarify the term of burst deletion as follows. Given an integer s ,

- a burst of consecutive deletions of length s [9], [27],

The research of G. Ge was supported by the National Key Research and Development Program of China under Grant 2020YFA0712100 and Grant 2018YFA0704703, the National Natural Science Foundation of China under Grant 11971325 and Grant 12231014, and Beijing Scholars Program.

The research of Y. Zhang was supported by the National Key Research and Development Program of China under Grant 2022YFA1004900 and Grant 2021YFA1001000, the National Natural Science Foundation of China under Grant 12001323, and by the Shandong Provincial Natural Science Foundation under Grant ZR2021YQ46.

Y. Sun and G. Ge are with the School of Mathematical Sciences, Capital Normal University, Beijing 100048, China (e-mail: 2200502135@cnu.edu.cn, gnge@zju.edu.cn).

Y. Zhang is with the Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, and also with the School of Cyber Science and Technology, Shandong University, Qingdao, Shandong 266237, China (e-mail: ywzhang@sdu.edu.cn).

[37], [39], refers to the error where exactly s consecutive symbols are deleted;

- a burst of consecutive deletions of length at most s [9], [30], [37], refers to the error where at most s consecutive symbols are deleted;
- a burst of nonconsecutive deletions of length at most s [37], also known as the localized deletions [5], [21], refers to the error where some symbols are deleted and the deleted coordinates are within a block of length at most s .

In this paper we only consider the first two types of errors, where the first is called a fixed length burst and the second is called a variable length burst. A series of works [6], [7], [9], [18], [19] have considered permutation codes and multi-permutation codes against burst deletions, for both stable and unstable types of errors and for both fixed length burst and variable length burst models. Among all the known results, so far the best permutation codes in S_n correcting a fixed length burst of stable deletions have redundancy $2 \log n + O(1)$, while the lower bound of the redundancy for such codes derived by a sphere-packing argument is only $\log n$ [9]. In this paper, we focus on burst stable deletions and one of our main results is to narrow the previous gap by constructing a code with redundancy $\log n + 2 \log \log n + O(1)$. The main challenge in this area is to retrieve the missing symbol on the first row of the array representation of a permutation. Since the alphabet size is n , the traditional method which simply applies a sum constraint modulo n on the first row will require $\log n$ bits of redundancy. One of our main contributions is to provide a novel strategy to retrieve the missing symbol with less redundancy. Using this strategy, we improve the redundancy of burst stable deletion correcting permutation codes. We then generalize our construction of permutation codes against fixed length burst stable deletions into variable length burst, and also their multi-permutation counterparts. Finally, we investigate the coding scheme in permutation codes and present a linear-time encoder correcting a single stable deletion with $\log n$ bits of redundancy.

The rest of this paper is organized as follows. Section II introduces the relevant definitions and notations used throughout the paper as well as the previous related results. In Section III we present a novel approach to retrieve the missing symbol on the first row of the array representation of a permutation or a multi-permutation, which plays a key role in our main constructions. In Section IV and Section V, we study permutation and multi-permutation codes against burst stable deletions, for fixed length burst and variable length burst, respectively. In Section VI, we present a linear-time encoder for single stable deletion correcting permutation codes. Finally Section VII concludes the paper.

II. PRELIMINARIES

We begin with introducing the relevant definitions and notations used throughout the paper. Let Σ_q denote the alphabet $\{0, 1, \dots, q-1\}$ with q symbols and Σ_q^n denote the set of all sequences of length n over Σ_q . Similarly, \mathbb{Z}^n denotes the set of vectors of length n over integers. A vector is always

denoted by bold letters, such as \mathbf{x} , and its i -th entry is denoted as x_i . For two positive integers i, j with $i \leq j$, the set of integers $\{i, i+1, \dots, j\}$ is denoted as $[i, j]$ and the set $[1, j] = \{1, 2, \dots, j\}$ is abbreviated as $[j]$. For two vectors $\mathbf{u} = (u_1, u_2, \dots, u_t)$ and $\mathbf{v} = (v_1, v_2, \dots, v_n)$, let (\mathbf{u}, \mathbf{v}) denote their *concatenation* $(u_1, u_2, \dots, u_t, v_1, v_2, \dots, v_n)$. If $t \leq n$ and there exist some integers $1 \leq i_1 < i_2 < \dots < i_t \leq n$ such that $u_j = v_{i_j}$ for all $1 \leq j \leq t$, then we say that \mathbf{u} is a *subsequence* of \mathbf{v} . Furthermore, if $i_{j+1} = i_j + 1$ for $1 \leq j \leq t-1$, then \mathbf{u} is called a *consecutive subsequence* of \mathbf{v} .

Let S_n be the symmetric group of order n . For any permutation $\sigma = (\sigma_1, \sigma_2, \dots, \sigma_n) \in S_n$, define $\sigma(\mathcal{I}) = \{\sigma_i : i \in \mathcal{I}\}$ for any subset $\mathcal{I} \subseteq [n]$. For any subset X of positive integers and a positive integer $k \notin X$, define $k(X) = k - |\{i \in X : i \leq k\}|$. Thus $k(X) = k'$ means that the positive integer k becomes the k' -th smallest positive integer in $\mathbb{Z}^+ \setminus X$. For example, if $X = \{2, 4, 6\}$ then $1(X) = 1, 3(X) = 2, 5(X) = 3$, and $k(X) = k - 3$ for $k \geq 7$.

As with most existing works concerning burst deletions, it is convenient to represent a vector in the form of an array. Given an integer s , we always assume that $n = st$ holds for some integer t . A vector $\sigma = (\sigma_1, \sigma_2, \dots, \sigma_n)$ is represented as the following $s \times t$ array

$$\begin{pmatrix} \sigma_1 & \sigma_{s+1} & \cdots & \sigma_{(t-1)s+1} \\ \sigma_2 & \sigma_{s+2} & \cdots & \sigma_{(t-1)s+2} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_s & \sigma_{2s} & \cdots & \sigma_n \end{pmatrix},$$

where the i -th row and j -th column of the array is denoted as $\sigma_{(s,i)}$ and $\sigma^{(s,j)}$, respectively.

Now we formally define the types of errors and the corresponding codes studied in this paper.

A. Permutation codes

Definition 2.1 (Stable deletions in permutations): Let $\sigma = (\sigma_1, \sigma_2, \dots, \sigma_n)$ be a permutation in S_n and \mathcal{I} be a subset of $[n]$ of size s . We say that σ suffers s *stable deletions* in \mathcal{I} and results in $\sigma' = (\sigma'_1, \sigma'_2, \dots, \sigma'_{n-s})$, if for all $k \in [1, n] \setminus \mathcal{I}$ and $i = k(\mathcal{I})$, we have $\sigma'_i = \sigma_k$.

Definition 2.2 (Unstable deletions in permutations): Let $\sigma = (\sigma_1, \sigma_2, \dots, \sigma_n)$ be a permutation in S_n and \mathcal{I} be a subset of $[n]$ of size s . We say that σ suffers s *unstable deletions* in \mathcal{I} and results in $\sigma' = (\sigma'_1, \sigma'_2, \dots, \sigma'_{n-s})$, if for all $k \in [1, n] \setminus \mathcal{I}$ and $i = k(\mathcal{I})$, we have $\sigma'_i = \sigma_k(\sigma(\mathcal{I}))$.

Example 2.3: Let $\sigma = (2, 4, 5, 1, 6, 3) \in S_6$ and let the set of deleted coordinates be $\mathcal{I} = \{2, 4\}$. In the model of stable deletions, the resultant vector is $(2, 5, 6, 3)$. In the model of unstable deletions, the resultant vector is $(1, 3, 4, 2)$.

Note that in the model of stable deletions the remaining symbols stay invariant and thus the resultant vector may no longer be a permutation, whereas in the model of unstable deletions the remaining symbols may change and form a new permutation in S_{n-s} based on their relative orders. This is also why the two terms were named as symbol-invariant and

permutation-invariant in [12]. In this paper we only consider stable deletions.

Definition 2.4 (Burst stable deletions in permutations): In Definition 2.1, if \mathcal{I} is an interval of length s of the form $\mathcal{I} = [i, i + s - 1]$ for some $i \in [n - s + 1]$, then we say that σ suffers a *burst stable deletion* of length s .

Immediately we have the following observation.

Observation 2.5: In the $s \times t$ array representation of a permutation $\sigma \in S_n$, a burst deletion of length s results in exactly one deletion on each row and the deleted coordinates are within at most two adjacent columns.

Definition 2.6: Let $\mathcal{B}_s(\sigma)$ be the set of all subsequences which are resulted from σ after s stable deletions. A code $\mathcal{C} \subseteq S_n$ is called an *s-SD permutation code*, if for any two distinct permutations $\sigma_1 \in \mathcal{C}, \sigma_2 \in \mathcal{C}$, it holds that $\mathcal{B}_s(\sigma_1) \cap \mathcal{B}_s(\sigma_2) = \emptyset$. That is, \mathcal{C} can correct exactly s stable deletions. Similarly we can define the following classes of codes:

- An *s-BSD permutation code* can correct a burst stable deletion of length s .
- An *$\leq s$ -BSD permutation code* can correct a burst stable deletion of length at most s .

B. Multi-permutation codes

A multi-permutation is a generalization of a permutation where each element appears multiple times that are known a priori.

Definition 2.7: Let n, w be positive integers such that $w \leq n$. A vector $\mathbf{r} = (r_1, r_2, \dots, r_w) \in \mathbb{Z}^w$ is called a *multiplicity vector* if $n = \sum_{i=1}^w r_i$ and $r_i \geq 1$ for $1 \leq i \leq w$. Let $M(n, \mathbf{r})$ denote the multiset

$$\underbrace{\{1, \dots, 1\}}_{r_1}, \underbrace{\{2, \dots, 2\}}_{r_2}, \dots, \underbrace{\{w, \dots, w\}}_{r_w}.$$

A *multi-permutation* $\sigma = (\sigma_1, \sigma_2, \dots, \sigma_n)$ is an arrangement of the elements of the multiset $M(n, \mathbf{r})$, i.e., the number i appears exactly r_i times in σ , $1 \leq i \leq w$. Let $S_n^{\mathbf{r}}$ denote the set of all multi-permutations on $M(n, \mathbf{r})$. In particular, if $n = wr$ and $r_1 = r_2 = \dots = r_w = r$, we say that the multiset and the corresponding multi-permutations are *regular*.

Terminologies regarding permutations can be naturally generalized for multi-permutations. For the interest of this paper we only list the following definitions.

Definition 2.8: Let $\mathbf{r} = (r_1, r_2, \dots, r_w)$ be a multiplicity vector. Let $\sigma = (\sigma_1, \sigma_2, \dots, \sigma_n)$ be a multi-permutation in $S_n^{\mathbf{r}}$ and let $\mathcal{I} = [i, i + s - 1]$ be an interval of length s for some $i \in [n - s + 1]$. We say that σ suffers a *burst stable deletion* of length s in \mathcal{I} , if the resultant vector is $(\sigma_1, \dots, \sigma_{i-1}, \sigma_{i+s}, \dots, \sigma_n)$.

Example 2.9: Let $\sigma = (3, 1, 3, 2, 2, 1, 2, 1, 3) \in S_9^{(3,3,3)}$ and $\mathcal{I} = [2, 4]$. If σ suffers a burst stable deletion of length 3 in \mathcal{I} , the resultant vector is $(3, 2, 1, 2, 1, 3)$.

Definition 2.10: Let \mathbf{r} be a multiplicity vector. We say a code $\mathcal{C} \subseteq S_n^{\mathbf{r}}$ is an *s-BSD multi-permutation code* if it can correct a burst stable deletion of length s , or an *$\leq s$ -BSD multi-permutation code* if it can correct a burst stable deletion of length at most s .

C. q-ary codes

For q -ary codes the deletion errors are naturally stable deletions.

Definition 2.11: Let $\mathbf{x} = (x_1, x_2, \dots, x_n)$ be a sequence in Σ_q^n and let $\mathcal{I} = [i, i + s - 1]$ be an interval of length s for some $i \in [n - s + 1]$. We say that \mathbf{x} suffers a *burst stable deletion* of length s in \mathcal{I} , if the resultant vector is $(x_1, \dots, x_{i-1}, x_{i+s}, \dots, x_n)$.

Definition 2.12: A code $\mathcal{C} \subseteq \Sigma_q^n$ is a *q-ary s-burst deletion correcting code* if it can correct a burst stable deletion of length s , or a *q-ary $\leq s$ -burst deletion correcting code* if it can correct a burst stable deletion of length at most s .

D. Previous results

In this subsection we present some known constructions of codes against deletion errors or burst deletion errors. The history of codes against deletion errors dates back to 1966, when Levenshtein [26] showed that the celebrated *Varshamov-Tenengol'ts (VT) codes* [45] are binary codes capable of correcting one deletion error. Recently, due to various applications in synchronization error-correction [17], [31], [32], DNA storage [13], [29], [46], and racetrack memories [8] et al., much progress has been made on binary or q -ary codes against multiple deletion errors [4], [14], [15], [20], [40]–[44].

To evaluate a code $\mathcal{C} \subseteq \mathcal{S}$, where the underlying set \mathcal{S} can be Σ_2^n, S_n , or $S_n^{\mathbf{r}}$, we either calculate its *size* $|\mathcal{C}|$ or analyze its *redundancy*, defined as $\log |\mathcal{S}| - \log |\mathcal{C}|$, where the logarithm base is 2.

The following list is not intended as a full survey of all the known results, but only focuses on the more related codes which will be the building blocks in our main construction. Related important terminologies and ideas are attached to specific constructions.

1) Single deletion correcting code:

- *Binary single deletion correcting code:* For $a \in \mathbb{Z}_{n+1}$, the *VT code* $\text{VT}_a(n)$ is defined as

$$\text{VT}_a(n) = \{\mathbf{x} \in \Sigma_2^n : \text{VT}(\mathbf{x}) \equiv a \pmod{(n+1)}\},$$

where $\text{VT}(\mathbf{x}) = \sum_{i=1}^n ix_i$ is called the *VT syndrome* of \mathbf{x} . Levenshtein [26] showed that $\text{VT}_a(n)$ is a binary single deletion correcting code.

- *q-ary single deletion correcting code:* For any sequence $\mathbf{x} = (x_1, x_2, \dots, x_n) \in \Sigma_q^n$, we define $\alpha(\mathbf{x}) = (\alpha(x_1), \alpha(x_2), \dots, \alpha(x_{n-1}))$, with $\alpha(x_i) = 1$ if $x_{i+1} \geq x_i$ and $\alpha(x_i) = 0$ otherwise, for all $i \in [n-1]$. The vector $\alpha(\mathbf{x})$ represents the relative order between consecutive symbols and is called the *signature* of \mathbf{x} . For $a \in \mathbb{Z}_n$ and $b \in \mathbb{Z}_q$, define

$$\text{VT}_{a,b}(n, q) = \{\mathbf{x} \in \Sigma_q^n : \alpha(\mathbf{x}) \in \text{VT}_a(n-1), \text{Sum}(\mathbf{x}) \equiv b \pmod{q}\},$$

where $\text{Sum}(\mathbf{x}) = \sum_{i=1}^n x_i$. Tenengol'ts [44] showed that $\text{VT}_{a,b}(n, q)$ is a q -ary single deletion correcting code.

- *Single stable deletion correcting permutation code:* Levenshtein [28] showed that permutations whose signature satisfies a VT constraint form a single stable deletion

correcting permutation code. Note that such a code can be seen as a subcode of $\text{VT}_{a,b}(n, q)$ restricting to S_n , with $q = n$ and $b = 1 + 2 + \dots + n$.

2) *P*-bounded single deletion correcting code (correcting a deletion given the additional knowledge of *P* consecutive coordinates which contain the erroneous coordinate):

- *Binary P*-bounded single deletion correcting code: For $a \in \mathbb{Z}_P$ and $b \in \mathbb{Z}_2$, the binary shifted VT code (SVT code) $\text{SVT}_{a,b}(n, P)$ is defined as

$$\text{SVT}_{a,b}(n, P) = \{ \mathbf{x} \in \Sigma_2^n : \text{VT}(\mathbf{x}) = a \pmod{P}, \\ \text{Sum}(\mathbf{x}) = b \pmod{2} \}.$$

Schoeny et al. [37] showed that $\text{SVT}_{a,b}(n, P)$ is a binary *P*-bounded single deletion correcting code. Note that compared to binary VT codes, in binary SVT codes the VT syndrome is computed modulo *P* instead of $n + 1$.

- *q*-ary *P*-bounded single deletion correcting code: For $a \in \mathbb{Z}_P$, $b \in \mathbb{Z}_2$, and $c \in \mathbb{Z}_q$, the *q*-ary SVT code $\text{SVT}_{a,b,c}(n, P, q)$ is defined as:

$$\text{SVT}_{a,b,c}(n, P, q) = \{ \mathbf{x} \in \Sigma_q^n : \text{Sum}(\mathbf{x}) = c \pmod{q}, \\ \alpha(\mathbf{x}) \in \text{SVT}_{a,b}(n-1, P) \}.$$

Schoeny et al. [38] showed that $\text{SVT}_{a,b,c}(n, P, q)$ is an *q*-ary *P*-bounded single deletion correcting code. Compared to *q*-ary VT codes, in *q*-ary SVT codes the signature of each codeword lies in a binary SVT code instead of a binary VT code.

3) *s*-burst deletion correcting code:

- *Binary s*-burst deletion correcting code: Schoeny et al. [37] proposed a construction of binary *s*-burst deletion correcting codes as follows. They represented each codeword in the form of an $s \times t$ array and constructed the codes for each row independently. The first row is a subcode of the VT code satisfying additional constraints known as the *run-length limited constraints*. A *run* in a sequence is a maximal consecutive subsequence consisting of the same symbol and a run-length limited constraint requires that the length of the longest run in a codeword is upper bounded by a predetermined value *P*. Under such constraints, by decoding the first row one can locate the deleted coordinate within a bounded interval of length *P* and thus provide additional information for the decoding of the remaining rows. Each of the remaining rows applies a $(P+1)$ -bounded single deletion correcting code. Here *P* is set to be of order $\log n$ so as to reduce the overall redundancy.
- *q*-ary *s*-burst deletion correcting code: Saeki et al. [39] generalized the binary codes in [37] by substituting the binary VT or SVT accessories therein as their *q*-ary versions, leading to a *q*-ary *s*-burst deletion correcting code.
- *s*-burst stable deletion correcting permutation code and multi-permutation code: Han et al. [18], [19] used an interleaving technique to construct *s*-burst stable deletion correcting permutation codes and multi-permutation codes. The current best result comes from Chee et

al. [9] with redundancy $2 \log n + O(1)$. They further generalized their construction of permutation codes to multi-permutation codes. Details of their construction are postponed to the next section, so as to bring out our main idea for improvements.

4) \leq_s -burst deletion correcting code:

- *Binary \leq_s -burst deletion correcting code*: Lenz et al. [30] followed the above framework of Schoeny et al. but encoded each row with a different strategy. Their main tool is an additional (\mathbf{p}, δ) -dense constraint which requires that any consecutive subsequence of length δ contains at least one pattern \mathbf{p} as a consecutive subsequence. A burst deletion may destroy or generate the occurrences of \mathbf{p} and thus by observing the pattern one can locate the burst error within a bounded interval. With this additional knowledge, several binary SVT codes are then applied for correction.
- *\leq_s -burst deletion correcting permutation code*: Chee et al. [9] constructed \leq_s -burst deletion correcting permutation codes by taking the intersection of their s' -burst stable deletion correcting permutation codes for all $1 \leq s' \leq s$, with some slight modifications.

III. RETRIEVE THE MISSING SYMBOL IN THE FIRST ROW OF THE ARRAY REPRESENTATION

By viewing a permutation as an $s \times \frac{n}{s}$ array, a burst stable deletion of length *s* results in exactly one stable deletion on each row. We briefly go over the construction of Chee et al. [9]. They imposed an *n*-ary VT constraint to the first row of the array, which would help retrieve the missing symbol in the first row of the array representation. As for permutations, this means the erroneous coordinate in the first row can be exactly located. Then the erroneous coordinates in the other rows are within two adjacent columns. Next, they recorded two values which add up the permutation rank (to be defined later in Definition 4.1) of the vector concatenated by two consecutive columns, where the two sums begin with the first and the second column, respectively. One of these two sums will contain the concatenation of the two erroneous columns and reveals its permutation rank. With the permutation rank and the known set of missing symbols, one can perform the correction.

Most redundancy ($2 \log n$ bits) of the codes above come from the *n*-ary VT constraint in the first row. The *n*-ary VT constraint contains two parts, the binary VT constraint on the signature and a sum constraint of all entries modulo *n*. While the first part seems inevitable, the second part can be improved, in the sense that the large alphabet size *n* can be significantly reduced if we have some additional constraint on the error pattern. Our key idea for improvements is to retrieve the missing symbol in the first row in a different way with less redundancy, consisting of the following three steps. For the rest of this paper we fix $P = \lceil \log \frac{4n}{s} \rceil$ and assume *n* is a multiple of $2Ps$.

- Step 1. Locate the erroneous coordinate of the first row of the array within an interval of length *P*.
- Step 2. Find a proper block of $2P$ consecutive columns which contains all the erroneous coordinates.

• Step 3. Retrieve the missing symbol of the first row, by analyzing the $2P$ consecutive columns from the previous step.

Step 1 can be guaranteed by applying some additional constraint on the first row. Since the techniques for Step 1 vary for permutation/multi-permutation codes against fixed/variable length burst, we leave it later to the next two sections when we introduce each specific code. In this section, we assume that we have already finished Step 1 and know the interval of length P containing the erroneous coordinate of the first row, and explain Steps 2-3 (which are two universal steps regardless of permutations/multi-permutations and fixed/variable burst stable deletions).

A. Step 2: Find $2P$ consecutive columns which contain all erroneous coordinates

Definition 3.1: For appropriate integers s , P , and i . Let $\sigma(s, P, i) \triangleq (\sigma^{(s, (i-1)P+1)}, \sigma^{(s, (i-1)P+2)}, \dots, \sigma^{(s, iP)})$, be the subarray of σ consisting of the columns indexed from $(i-1)P+1$ to iP , i.e.

$$\sigma(s, P, i) = \begin{pmatrix} \sigma_{(i-1)Ps+1} & \sigma_{((i-1)P+1)s+1} & \cdots & \sigma_{(iP-1)s+1} \\ \sigma_{(i-1)Ps+2} & \sigma_{((i-1)P+1)s+2} & \cdots & \sigma_{(iP-1)s+2} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{((i-1)P+1)s} & \sigma_{((i-1)P+2)s} & \cdots & \sigma_{iPs} \end{pmatrix}.$$

Suppose the erroneous coordinate of the first row is in the interval $\mathcal{I} = [i, j]$ with $j - i + 1 \leq P$. Then the erroneous coordinates in the other rows are within the columns indexed by $[i-1, j]$ if $i \geq 2$, or $[1, j]$ if $i = 1$. According to Definition 3.1, there is at least one block $\varsigma = (\sigma(s, P, k), \sigma(s, P, k+1))$ containing the erroneous columns, as shown in Figure 1.

Moreover, all entries in the block ς are known (since it is the complement of the set of entries on the other non-erroneous columns).

B. Step 3: Retrieve the missing symbol of the first row

Clearly, to retrieve the missing symbol of the first row, one may simply apply a sum constraint modulo n for the first row $\sigma_{(s,1)}$. However, it would still require $\log n$ bits of redundancy. Below we will show that, with the help of Step 2, the missing symbol of the first row can be retrieved with less redundancy. We need the following definition.

Definition 3.2: Let $\mathbf{u} = (u_1, u_2, \dots, u_n)$ be a vector with integer entries. Define $\beta(\mathbf{u}) = (\beta(\mathbf{u})_1, \beta(\mathbf{u})_2, \dots, \beta(\mathbf{u})_n) \in S_n$ to be the permutation induced by \mathbf{u} as follows:

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

That is, we rank the entries of \mathbf{u} in an increasing order and the repeated symbols are ordered according to their appearances. $\beta(\mathbf{u})$ is called the *permutation projection* of \mathbf{u} and $\beta(\mathbf{u})_i$ is the *order* of the symbol u_i in \mathbf{u} .

Example 3.3: Let $\mathbf{u} = (6, 2, 5, 1, 8, 4)$, $\mathbf{v} = (1, 1, 2, 2, 1, 2)$. Then $\beta(\mathbf{u}) = (5, 2, 4, 1, 6, 3)$ and $\beta(\mathbf{v}) = (1, 2, 4, 5, 3, 6)$.

Lemma 3.4: Let $\mathcal{J} = \{\varsigma_1, \varsigma_2, \dots, \varsigma_{2Ps}\} \subseteq [n]$ be a set of $2Ps$ distinct elements and $\varsigma = (\varsigma_1, \varsigma_2, \dots, \varsigma_{2Ps})$ be

a vector written in an $s \times 2P$ array. Suppose we know $\text{Sum}(\beta(\varsigma)_{(s,1)}) \equiv c \pmod{2Ps}$ with $c \in \mathbb{Z}_{2Ps}$. If ς suffers a burst stable deletion of length s , then we can retrieve the missing symbol of the first row given the knowledge of the set \mathcal{J} .

Proof: Assume ς suffers a burst stable deletion of length s and the resultant vector is $\mathbf{v} = (v_1, v_2, \dots, v_{(2P-1)s})$. Since the entries of ς are distinct, its permutation projection naturally induces a bijection $f_\beta : \{\varsigma_1, \varsigma_2, \dots, \varsigma_{2Ps}\} \rightarrow [2Ps]$ where $f_\beta(\varsigma_i) = \beta(\varsigma)_i$. Let $\tilde{\mathbf{v}} = (f_\beta(v_1), f_\beta(v_2), \dots, f_\beta(v_{(2P-1)s}))$. Then $\tilde{\mathbf{v}}$ can be seen as a resultant vector of $\beta(\varsigma)$ suffering a burst stable deletion of length s . Consider their first rows, $\tilde{\mathbf{v}}_{(s,1)} \in \mathcal{B}_1(\beta(\varsigma)_{(s,1)})$. Therefore, as long as we know that $\text{Sum}(\beta(\varsigma)_{(s,1)}) \equiv c \pmod{2Ps}$, the missing symbol of $\varsigma_{(s,1)}$ can be retrieved as the preimage $f_\beta^{-1}(c - \text{Sum}(\tilde{\mathbf{v}}_{(s,1)}) \pmod{2Ps})$. ■

Example 3.5: Assume $\mathcal{J} = \{1, 3, 4, 9, 12, 13, 15, 16\}$, $\varsigma = (4, 9, 1, 12, 3, 15, 16, 13)$, $\beta(\varsigma) = (3, 4, 1, 5, 2, 7, 8, 6)$, and $\text{Sum}(\beta(\varsigma)_{(2,1)}) \equiv 6 \pmod{8}$. Suppose ς suffers a burst stable deletion of length 2 and results in $\mathbf{v} = (4, 9, 1, 15, 16, 13)$. With the knowledge of the set \mathcal{J} , we have $\tilde{\mathbf{v}} = (3, 4, 1, 7, 8, 6)$, $\tilde{\mathbf{v}}_{(2,1)} = (3, 1, 8)$, and $\text{Sum}(\tilde{\mathbf{v}}_{(2,1)}) \equiv 4 \pmod{8}$. The missing symbol of $\beta(\varsigma)_{(2,1)}$ is thus 2 and the missing symbol of $\varsigma_{(2,1)}$ is $f_\beta^{-1}(2) = 3$.

Remark 3.6: Note that Lemma 3.4 also holds when the elements $\{\varsigma_1, \varsigma_2, \dots, \varsigma_{2Ps}\}$ in \mathcal{J} are not distinct. While the bijection f_β does not make sense anymore, there is no problem in finding the missing symbol t in $\beta(\varsigma)_{(s,1)}$ and retrieve the missing symbol of $\varsigma_{(s,1)}$ to be the t -th smallest element in the (multi-set) \mathcal{J} . In fact, let

$$\mathcal{J} = \underbrace{\{a_1, \dots, a_1\}}_{r_1}, \underbrace{\{a_2, \dots, a_2\}}_{r_2}, \dots, \underbrace{\{a_w, \dots, a_w\}}_{r_w}$$

with $\sum_{j=1}^w r_j = 2Ps$ and $a_1 < a_2 < \dots < a_w$. If $\sum_{j=1}^i r_j + 1 \leq t \leq \sum_{j=1}^{i+1} r_j$ for some i , then the t -th smallest element in the multi-set \mathcal{J} is a_{i+1} .

Lemma 3.4 suggests that, if in Step 2 we find a proper block ς containing the $P+1$ erroneous columns, then the missing symbol of the first row can be retrieved as long as we know the value c such that $\text{Sum}(\beta(\varsigma)_{(s,1)}) \equiv c \pmod{2Ps}$. The next part describes how to derive such a parameter c .

Definition 3.7: Let $\mathcal{C}_{c_1, c_2}^{\text{retrieve}}(n, s, P)$, with $c_1, c_2 \in \mathbb{Z}_{2Ps}$, be the set of permutations or multi-permutations, such that each codeword σ satisfies the following constraints

$$\begin{cases} \sum_{i=1}^{n/2Ps} \text{Sum}(\beta(\sigma(s, P, 2i-1), \sigma(s, P, 2i))_{(s,1)}) \\ \equiv c_1 \pmod{2Ps}, \\ \sum_{i=1}^{n/2Ps} \text{Sum}(\beta(\sigma(s, P, 2i), \sigma(s, P, 2i+1))_{(s,1)}) \\ \equiv c_2 \pmod{2Ps}, \end{cases} \quad (1)$$

where $\sigma(s, P, n/Ps+1) = \sigma(s, P, 1)$.

Remark 3.8: In Definition 3.7 we partite σ into $\frac{n}{Ps}$ disjoint subarrays, where $\frac{n}{Ps}$ is an even integer. We combine two subarrays $\sigma(s, P, 2i-1), \sigma(s, P, 2i)$ together and consider the first row of its permutation projection $\mathbf{v}_{2i-1, 2i} \triangleq$

$$\begin{aligned}
\boldsymbol{\sigma} &= \begin{pmatrix} \sigma_1 & \cdots & \sigma_{(k-1)Ps+1} & \cdots & \sigma_{(i-2)s+1} & \cdots & \sigma_{(j-1)s+1} & \cdots & \sigma_{((k+1)P-1)s+1} & \cdots & \sigma_{(t-1)s+1} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \sigma_s & \cdots & \sigma_{((k-1)P+1)s} & \cdots & \sigma_{(i-1)s} & \cdots & \sigma_{js} & \cdots & \sigma_{(k+1)Ps} & \cdots & \sigma_n \end{pmatrix} \\
&\quad \downarrow \\
\boldsymbol{\varsigma} = (\boldsymbol{\sigma}(s, P, k), \boldsymbol{\sigma}(s, P, k+1)) &= \begin{pmatrix} \sigma_{(k-1)Ps+1} & \cdots & \sigma_{(i-2)s+1} & \cdots & \sigma_{(j-1)s+1} & \cdots & \sigma_{((k+1)P-1)s+1} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \sigma_{((k-1)P+1)s} & \cdots & \sigma_{(i-1)s} & \cdots & \sigma_{js} & \cdots & \sigma_{(k+1)Ps} \end{pmatrix}
\end{aligned}$$

Fig. 1. Find $2P$ consecutive columns which contain all erroneous coordinates.

$\beta(\boldsymbol{\sigma}(s, P, 2i-1), \boldsymbol{\sigma}(s, P, 2i))_{(s,1)}$. Regard $\mathbf{v}_{2i-1,2i}$ as a q -ary vector of length $2P$ where $q = 2Ps$ and we calculate the sum of its entries. In the first equation of Equation (1) we do the summation over $\{\mathbf{v}_{2i-1,2i} : 1 \leq i \leq \frac{n}{2Ps}\}$. The second equation is a similar summation over $\{\mathbf{v}_{2i,2i+1} : 1 \leq i \leq \frac{n}{2Ps}\}$.

Lemma 3.9: Suppose $\boldsymbol{\sigma}$ suffers a burst stable deletion of length s and we already find $\boldsymbol{\varsigma} = (\boldsymbol{\sigma}(s, P, k), \boldsymbol{\sigma}(s, P, k+1))$, a block of $2P$ consecutive columns containing all erroneous coordinates. Moreover, if $\boldsymbol{\sigma} \in \mathcal{C}_{c_1, c_2}^{\text{retrieve}}(n, s, P)$, then the value c such that $\text{Sum}(\beta(\boldsymbol{\varsigma})_{(s,1)}) \equiv c \pmod{2Ps}$ can be determined.

Proof: To find the value c such that $\text{Sum}(\beta(\boldsymbol{\varsigma})_{(s,1)}) \equiv c \pmod{2Ps}$, we turn to either the first or the second equation in Equation (1), depending on the parity of k . WLOG let k be odd, then by the first equation, we have

$$\begin{aligned}
c &= \text{Sum}(\beta(\boldsymbol{\varsigma})_{(s,1)}) \\
&\equiv c_1 - \sum_{i \neq (k+1)/2} \text{Sum}(\mathbf{v}_{2i-1,2i}) \pmod{2Ps},
\end{aligned}$$

where $\mathbf{v}_{2i-1,2i} \triangleq \beta(\boldsymbol{\sigma}(s, P, 2i-1), \boldsymbol{\sigma}(s, P, 2i))_{(s,1)}$, and thus the value c is determined. ■

Example 3.10: Consider the set $\mathcal{C}_{6,2}^{\text{retrieve}}(16, 2, 2)$, suppose $\boldsymbol{\sigma} = (7, 8, 2, 5, 4, 9, 1, 12, 3, 15, 16, 13, 14, 6, 11, 10)$ (from the set) suffers a burst stable deletion of length 2 and the resultant vector is $\boldsymbol{\tau} = (7, 8, 2, 5, 4, 9, 1, 15, 16, 13, 14, 6, 11, 10)$. In addition, suppose that we have already known that the erroneous coordinate in the first row belongs to the interval $[4, 5]$. Then we can find the block $\boldsymbol{\varsigma} = (\boldsymbol{\sigma}(2, 2, 2), \boldsymbol{\sigma}(2, 2, 3))$ containing the three possible erroneous columns, and calculate the parameter c such that $\text{Sum}(\beta(\boldsymbol{\varsigma})_{(2,1)}) \equiv c \pmod{8}$, by $c = 2 - \text{Sum}(\beta(\boldsymbol{\sigma}(2, 2, 4), \boldsymbol{\sigma}(2, 2, 1))_{(2,1)}) \equiv 6 \pmod{8}$.

C. Summary of this section

In this section, we introduce a different strategy to retrieve the missing symbol of $\boldsymbol{\sigma}_{(s,1)}$. We summarize this section into the following theorem, which will be the key building block for our codes in the next two sections.

Theorem 3.11: Consider a permutation (multi-permutation) $\boldsymbol{\sigma} \in \mathcal{C}_{c_1, c_2}^{\text{retrieve}}(n, s, P)$ and assume that $\boldsymbol{\sigma}$ suffers a burst stable deletion of length s . If the erroneous coordinate of $\boldsymbol{\sigma}_{(s,1)}$ is within a known interval of length P , then the missing symbol of $\boldsymbol{\sigma}_{(s,1)}$ can be exactly retrieved.

Proof: Since the erroneous coordinate of $\boldsymbol{\sigma}_{(s,1)}$ is within a known interval of length P , the erroneous coordinates on

the other rows are within some $P+1$ columns. By Step 2, we can find a proper block $\boldsymbol{\varsigma} = (\boldsymbol{\sigma}(s, P, k), \boldsymbol{\sigma}(s, P, k+1))$ of $2P$ consecutive columns containing all erroneous coordinates. By Lemma 3.9 in Step 3, the value c such that $\text{Sum}(\beta(\boldsymbol{\varsigma})_{(s,1)}) \equiv c \pmod{2Ps}$ can be determined since $\boldsymbol{\sigma} \in \mathcal{C}_{c_1, c_2}^{\text{retrieve}}(n, s, P)$. Then by Lemma 3.4 in Step 3, the missing symbol of $\boldsymbol{\varsigma}_{(s,1)}$ (and thus $\boldsymbol{\sigma}_{(s,1)}$) can be retrieved. ■

IV. s -BSD PERMUTATION/MULTI-PERMUTATION CODES

In this section we consider the fixed length burst model and present our constructions of s -BSD permutation and multi-permutation codes. Our code has $\log n + 2 \log \log n + O(1)$ bits of redundancy, which outperforms the previously best known result from [9] with $2 \log n + O(1)$ bits of redundancy. The following definition is needed.

Definition 4.1: For two distinct permutations $\boldsymbol{\sigma}, \boldsymbol{\pi} \in S_n$, define the partial order $\boldsymbol{\sigma} \preceq \boldsymbol{\pi}$ if there exists some $1 \leq j \leq n$ such that $\sigma_i = \pi_i$ for $1 \leq i < j$ and $\sigma_j < \pi_j$. This is a total order in S_n known as the *lexicographic order*. Let $\ell(\boldsymbol{\sigma})$ be the *lexicographic rank* of $\boldsymbol{\sigma} \in S_n$, i.e., $\boldsymbol{\sigma}$ is the $\ell(\boldsymbol{\sigma})$ -th smallest permutation in the poset (S_n, \preceq) . For a vector \mathbf{u} of length n with positive integer entries, recall that $\beta(\mathbf{u})$ is the permutation projection of \mathbf{u} and we define the *permutation rank* of \mathbf{u} as $\mu(\mathbf{u}) = \ell(\beta(\mathbf{u}))$.

Remark 4.2: Note that we can recover an integer vector \mathbf{u} given the knowledge of its permutation rank $\mu(\mathbf{u})$ and the multiset of all its entries $\{u_i : 1 \leq i \leq n\}$. For example, if a vector $\mathbf{u} = (u_1, u_2, u_3, u_4)$ has permutation rank $\mu(\mathbf{u}) = 2$, then $\beta(\mathbf{u}) = (1, 2, 4, 3)$. Furthermore, if $\{u_1, u_2, u_3, u_4\} = \{1, 3, 5, 6\}$, then $\mathbf{u} = (1, 3, 6, 5)$.

We now present our codes against fixed length burst stable deletions. Subsection IV-A deals with permutations and Subsection IV-B deals with multi-permutations.

A. s -BSD permutation codes

Definition 4.3: Consider a permutation $\boldsymbol{\sigma} \in S_n$ written in an array of size $s \times \frac{n}{s}$. Let $\alpha(\boldsymbol{\sigma}_{(s,1)})$ be the signature of its first row. If $\alpha(\boldsymbol{\sigma}_{(s,1)})$ is a binary vector with maximal runlength at most $P-1$, then we say $\boldsymbol{\sigma}$ is a *good permutation*.

Lemma 4.4: The number of good permutations is at least $\frac{n!}{2}$.

Proof: We prove the lemma by a probabilistic analysis. Uniformly choose a random permutation $\boldsymbol{\sigma}$ in S_n . Then, for each $i \in [1, \frac{n}{s} - P]$, let \mathbb{E}_i be the event that

$(\alpha(\sigma_{(s,1)})_i, \alpha(\sigma_{(s,1)})_{i+1}, \dots, \alpha(\sigma_{(s,1)})_{i+P-1})$ is either $\mathbf{0}^P$ or $\mathbf{1}^P$, i.e., either $\sigma_{(i-1)s+1} > \sigma_{is+1} > \dots > \sigma_{(i+P-1)s+1}$ or $\sigma_{(i-1)s+1} < \sigma_{is+1} < \dots < \sigma_{(i+P-1)s+1}$. The probability of \mathbb{E}_i can be calculated by

$$\Pr(\mathbb{E}_i) = \frac{2}{(P+1)!} \leq \frac{2}{2^P} \leq \frac{s}{2n}.$$

By the union bound, the probability of the event that σ is not a good permutation is upper bounded by

$$\begin{aligned} & \Pr(\sigma \text{ is not a good permutation}) \\ & \leq \sum_{i=1}^{\frac{n}{s}-P} \Pr(\mathbb{E}_i) \leq \frac{s}{2n} \left(\frac{n}{s} - P \right) \leq \frac{1}{2}, \end{aligned}$$

and thus the number of good permutations is at least $\frac{n!}{2}$. ■

Following Lemma 4.4, we consider good permutations σ with $\alpha(\sigma_{(s,1)})$ following a binary VT constraint. When a burst stable deletion of length s occurs in σ , the first row suffers exactly one stable deletion and so does its signature. By decoding the first row of its signature, the erroneous coordinate of $\alpha(\sigma_{(s,1)})$ is bounded in an interval of length $P-1$. Now we explain the relation of the erroneous coordinate between a sequence and its signature.

Lemma 4.5 (Lemma 2, [39]): For a vector \mathbf{x} of length n , either $\alpha(\mathbf{x}_{[n] \setminus \{i\}}) = \alpha(\mathbf{x}_{[n-1] \setminus \{i-1\}})$ or $\alpha(\mathbf{x}_{[n] \setminus \{i\}}) = \alpha(\mathbf{x}_{[n-1] \setminus \{i\}})$ holds. That is, when the erroneous coordinate of a sequence is i , the erroneous coordinate of its signature is either $i-1$ or i .

By Lemma 4.5 we have the following lemma.

Lemma 4.6: If a good permutation σ suffers a burst deletion of length s and $\alpha(\sigma_{(s,1)})$ follows a binary VT constraint, the erroneous coordinate of $\sigma_{(s,1)}$ can be located in an interval of length P .

Up till now, we have used good permutations to finish Step 1 mentioned in the previous section. Then we can proceed with Steps 2 and 3 to retrieve the missing symbol of $\sigma_{(s,1)}$. Afterwards, some other constraint on the first row can help us exactly locate the erroneous coordinate of $\sigma_{(s,1)}$. Then the erroneous coordinates on the other rows are within two adjacent columns. The correction of these two columns follows the same way as in [9] and is rephrased here for completeness.

Definition 4.7: Let $\mathcal{C}_{d_1, d_2}^{\text{recover}}(n, s)$, with $d_1, d_2 \in \mathbb{Z}_{(2s)!}$, be the set of all permutations $\sigma \in S_n$ which satisfies the following constraints

$$\begin{cases} \sum_{i=1}^{n/2s} \mu(\sigma^{(s, 2i-1)}, \sigma^{(s, 2i)}) = d_1 \pmod{(2s)!} \\ \sum_{i=1}^{n/2s} \mu(\sigma^{(s, 2i)}, \sigma^{(s, 2i+1)}) = d_2 \pmod{(2s)!} \end{cases}, \quad (2)$$

where $\sigma^{(s, n/2s+1)} = \sigma^{(s, 1)}$.

Lemma 4.8: Suppose we have located all erroneous coordinates in two adjacent columns. If $\sigma \in \mathcal{C}_{d_1, d_2}^{\text{recover}}(n, s)$, then the two erroneous columns can be corrected by its permutation rank derived from Equation (2) and the set of its entries.

Example 4.9: Consider the set $\mathcal{C}_{2, 3}^{\text{recover}}(16, 2)$, suppose $\sigma = (7, 8, 2, 5, 4, 9, 1, 12, 3, 15, 16, 13, 14, 6, 11, 10)$ (from the set) suffers a burst stable deletion of length 2. If the resultant

vector is $\tau = (7, 8, 2, 5, 4, 9, 1, 15, 16, 13, 14, 6, 11, 10)$ and the erroneous coordinate of the first row is 5. All we need is to recover the columns indexed by 4 and 5 and it can be done as follows. Firstly, we know that $\sigma^{(2,1)} = (7, 8)$, $\sigma^{(2,2)} = (2, 5)$, $\sigma^{(2,3)} = (4, 9)$, $\sigma^{(2,6)} = (16, 13)$, $\sigma^{(2,7)} = (14, 6)$, $\sigma^{(2,8)} = (11, 10)$, the four elements in $\sigma^{(2,4)}$ and $\sigma^{(2,5)}$ are $\{1, 3, 12, 15\}$. Then by Equation (2) we obtain $\mu(\sigma^{(2,4)}, \sigma^{(2,5)}) = 3 - \sum_{j \in \{1, 3, 4\}} \mu(\sigma^{(2,2j)}, \sigma^{(2, 2j+1)}) = 3 \pmod{24}$. Then by Remark 4.2, the permutation projection is $\beta(\sigma^{(2,4)}, \sigma^{(2,5)}) = (1, 3, 2, 4)$ and thus $(\sigma^{(2,4)}, \sigma^{(2,5)}) = (1, 12, 3, 15)$.

Summing up the above, the full construction of our s -BSD permutation codes is as follows.

Construction 4.10: Let $a \in \mathbb{Z}_{n/s}$, $c_1, c_2 \in \mathbb{Z}_{2Ps}$, and $d_1, d_2 \in \mathbb{Z}_{(2s)!}$. Define the code $\mathcal{C}_s^1 \triangleq \mathcal{C}_s^1(n; a; c_1, c_2; d_1, d_2)$ as

$$\begin{aligned} \mathcal{C}_s^1 = \{ & \sigma \in S_n : \alpha(\sigma_{(s,1)}) \in VT_a(n/s-1), \sigma \text{ is good,} \\ & \sigma \in \mathcal{C}_{c_1, c_2}^{\text{retrieve}}(n, s, P) \cap \mathcal{C}_{d_1, d_2}^{\text{recover}}(n, s)\}. \end{aligned}$$

Theorem 4.11: The permutation code $\mathcal{C}_s^1(n; a; c_1, c_2; d_1, d_2)$ is an s -BSD permutation code over S_n . Moreover, by choosing proper parameters there is an s -BSD permutation code with redundancy at most $\log n + 2 \log \log n + O(1)$.

Proof: Suppose $\sigma = (\sigma_1, \sigma_2, \dots, \sigma_n) \in \mathcal{C}_s^1$ suffers a burst stable deletion of length s . Since $\alpha(\sigma_{(s,1)}) \in VT_a(n/s-1)$ and σ is a good permutation, by Lemma 4.6 the erroneous coordinate of $\sigma_{(s,1)}$ can be located in an interval of length at most P . Moreover, since $\sigma \in \mathcal{C}_{c_1, c_2}^{\text{retrieve}}(n, s, P)$, by Theorem 3.11 we can retrieve the missing symbol of $\sigma_{(s,1)}$. Note that the erroneous coordinate of $\alpha(\sigma_{(s,1)})$ lies in a consecutive run of 0's or 1's, which means that the erroneous coordinate of $\sigma_{(s,1)}$ lies in a monotone decreasing sequence or a monotone increasing sequence. In such a sequence, the coordinate of the known missing symbol is certainly unique. Therefore, the erroneous coordinate of $\sigma_{(s,1)}$ can be exactly located. Finally, the two erroneous columns can be corrected by Lemma 4.8.

By the pigeon-hole principal, there exist parameters $a \in \mathbb{Z}_{n/s}$, $c_1, c_2 \in \mathbb{Z}_{2Ps}$, $d_1, d_2 \in \mathbb{Z}_{(2s)!}$ such that the redundancy of \mathcal{C}_s^1 is at most

$$\begin{aligned} & \log 2 + \log \frac{n}{s} + 2 \log(2Ps) + 2 \log(2s)! \\ & = \log n + 2 \log \log n + O(1), \end{aligned}$$

where the first term is the 1 bit redundancy due to the size of good permutations (Lemma 4.4). ■

Example 4.12: Consider the set $\mathcal{C}_2^1(16; 3; 6, 2; 2, 3)$, suppose $\sigma = (7, 8, 2, 5, 4, 9, 1, 12, 3, 15, 16, 13, 14, 6, 11, 10)$ (from the set) suffers a burst stable deletion of length 2 and the resultant vector is $\tau = (7, 8, 2, 5, 4, 9, 1, 15, 16, 13, 14, 6, 11, 10)$. Firstly, since $\alpha(\sigma_{(2,1)}) \in VT_3(7)$, we can use a VT decoder to recover $\alpha(\sigma_{(2,1)}) = (0, 1, 0, 1, 1, 0, 0)$ from $\alpha(\tau_{(2,1)}) = (0, 1, 0, 1, 0, 0)$, and locate the erroneous coordinate in the first row in the interval [4, 5]. Then we find a proper block $\varsigma = (\sigma(2, 2), \sigma(2, 3))$ and by Example 3.10, we obtain $\text{Sum}(\beta(\varsigma)_{(2,1)}) \equiv 6 \pmod{8}$. By Example 3.5, we retrieve the missing symbol of $\sigma_{(s,1)}$ to be 3. Now we use an n -ary VT decoder to recover $\sigma_{(2,1)}$ and locate the exact erroneous coordinate. Finally, all erroneous coordinates are located in

two adjacent columns and we recover these two columns by Example 4.9, which completes the decoding process.

B. s -BSD multi-permutation codes

Now we consider s -BSD multi-permutation codes in S_n^r , where $\mathbf{r} = (r_1, r_2, \dots, r_w)$ is the given multiplicity vector. Similarly as the permutation codes, we can recover the first row of a multi-permutation $\sigma \in S_n^r$. However note that in a multi-permutation there are repeated entries, and thus recovering $\sigma_{(s,1)}$ can only help us locate the erroneous coordinate within an interval of length r in the worst case, where $r = \max_{1 \leq i \leq w} r_i$. Thus, all the erroneous coordinates in the other rows are within at most $r+1$ columns. To deal with this issue we can reconsider σ in an array of size $s(r+1) \times \frac{n}{s(r+1)}$ and then all the erroneous coordinates are within at most two adjacent columns in this new array, and the correction of these two columns can be done in a similar way as Lemma 4.8.

In the constructions below we start with the case where $n = wr$ and $r_i = r$ for all $1 \leq i \leq w$ (i.e., \mathbf{r} is regular), and then for the irregular case. Moreover, w is assumed to be even for simplicity. Recall that we have fixed the parameter $P = \lceil \log \frac{4n}{s} \rceil$. First we view a multi-permutation as an array of size $s \times \frac{n}{s}$. We say that a multi-permutation σ is *good* if $\alpha(\sigma_{(s,1)})$ is a binary vector with maximal runlength at most $P-1$. Similarly, by a probabilistic analysis, the number of good multi-permutations is at least half the size of S_n^r .

Lemma 4.13: The number of good multi-permutations is at least $\frac{n!}{2^{(r!)^w}}$ when $P+1 \geq 2^{2r}$.

Proof: Uniformly choose a random multi-permutation $\sigma \in S_n^r$. For each $i \in [1, \frac{n}{s} - P]$, let \mathbb{E}_i be the event that $(\alpha(\sigma_{(s,1)})_i, \alpha(\sigma_{(s,1)})_{i+1}, \dots, \alpha(\sigma_{(s,1)})_{i+P-1})$ is either $\mathbf{0}^P$ or $\mathbf{1}^P$, i.e., either $\sigma_{(i-1)s+1} \geq \sigma_{is+1} \geq \dots \geq \sigma_{(i+P-1)s+1}$ or $\sigma_{(i-1)s+1} \leq \sigma_{is+1} \leq \dots \leq \sigma_{(i+P-1)s+1}$. For simplicity, we assume that $(P+1)$ is a multiple of r , and the probability of \mathbb{E}_i can be calculated by

$$\begin{aligned} \Pr(\mathbb{E}_i) &= \frac{\sum_{\substack{t_1+\dots+t_w=P+1 \\ 0 \leq t_1, \dots, t_w \leq r}} 2}{\sum_{\substack{t_1+\dots+t_w=P+1 \\ 0 \leq t_1, \dots, t_w \leq r}} \prod_{1 \leq j \leq w} \frac{(P+1)!}{t_j!}} \\ &\stackrel{(*)}{\leq} \frac{2}{2^{P-2r}(P+1)} \\ &\stackrel{(**)}{\leq} \frac{1}{2^{P-1}} \leq \frac{s}{2n}. \end{aligned}$$

Note that for any $t_1, t_2 \in [0, r]$, we have

$$(t_1)!(t_2)! \leq \begin{cases} (t_1+t_2)!, & \text{if } t_1+t_2 \leq r; \\ r!(t_1+t_2-r)!, & \text{otherwise.} \end{cases}$$

Thus, $(*)$ holds since $\prod_{1 \leq j \leq w} t_j! \leq (r!)^{\frac{P+1}{r}} \leq (2r)!r^{P-2r}$ for any choice of integers t_1, \dots, t_w such that $0 \leq t_1, \dots, t_w \leq r$ and $t_1 + \dots + t_w = P+1$, and $\frac{(P+1)!}{(2r)!r^{P-2r}} \geq 2^{P-2r}(P+1)$. $(**)$ holds when $P+1 \geq 2^{2r}$.

By the union bound, the probability of the event that σ is not a good multi-permutation is upper bounded by

$$\begin{aligned} \Pr(\sigma \text{ is not a good multi-permutation}) \\ \leq \sum_{i=1}^{\frac{n}{s}-P} \Pr(\mathbb{E}_i) \leq \frac{s}{2n} \left(\frac{n}{s} - P \right) \leq \frac{1}{2}, \end{aligned}$$

and thus the number of good multi-permutations is at least $\frac{n!}{2^{(r!)^w}}$. ■

The full construction of our s -BSD multi-permutation codes is as follows. Note that the code $\mathcal{C}_{d_1, d_2}^{\text{recover}}(n, s)$ in Definition 4.7 can be naturally generalized to multi-permutations.

Construction 4.14: Let $n = rw = s(r+1)t$ where t is even. Let $a \in \mathbb{Z}_{n/s}$, $c_1, c_2 \in \mathbb{Z}_{2Ps}$, and $d_1, d_2 \in \mathbb{Z}_{(2sr)!}$. Define the code $\mathcal{C}_s^2 \triangleq \mathcal{C}_s^2(n; a; c_1, c_2; d_1, d_2)$ as

$$\begin{aligned} \mathcal{C}_s^2 = \{ \sigma \in S_n^r : \alpha(\sigma_{(s,1)}) \in VT_a(n/s-1), \sigma \text{ is good,} \\ \sigma \in \mathcal{C}_{c_1, c_2}^{\text{retrieve}}(n, s, P) \cap \mathcal{C}_{d_1, d_2}^{\text{recover}}(n, s(r+1)) \}. \end{aligned}$$

Theorem 4.15: The code \mathcal{C}_s^2 is an s -BSD multi-permutation code over S_n^r . Moreover, by choosing proper parameters there is an s -BSD multi-permutation code over S_n^r with redundancy at most $\log n + 2 \log \log n + O(1)$.

Proof: Suppose $\sigma = (\sigma_1, \sigma_2, \dots, \sigma_n) \in \mathcal{C}_s^2$ suffers a burst stable deletion of length s . Since $\alpha(\sigma_{(s,1)}) \in VT_a(n/s-1)$ and σ is a good multi-permutation, by Lemma 4.6 the erroneous coordinate of $\sigma_{(s,1)}$ can be located in an interval of length at most P . Moreover, since $\sigma \in \mathcal{C}_{c_1, c_2}^{\text{retrieve}}(n, s, P)$, by Theorem 3.11 we can retrieve the missing symbol of $\sigma_{(s,1)}$. By an n -ary VT decoder, we can recover $\sigma_{(s,1)}$. However, due to the existence of repeated entries in $\sigma_{(s,1)}$, recovering $\sigma_{(s,1)}$ only helps locate the erroneous coordinate within an interval of length r , in the worst case. Finally, by viewing the multi-permutation in an array of size $s(r+1) \times t$, all the erroneous coordinates are within two adjacent columns. Since $\sigma \in \mathcal{C}_{d_1, d_2}^{\text{recover}}(n, s(r+1))$, the correction can be done in a similar way as Lemma 4.8.

The code redundancy can be calculated by the pigeon-hole principle immediately. ■

To construct codes for irregular multi-permutations, simply choose $r_{\max} = \max_{i \in [w]} r_i$ and apply the above construction.

Corollary 4.16: There exists an s -BSD multi-permutation code in S_n^r (here \mathbf{r} could be irregular) with redundancy at most $\log n + 2 \log \log n + O(1)$.

V. $\leq s$ -BSD PERMUTATION/MULTI-PERMUTATION CORRECTING CODES

Now we move on to the variable length burst. A trivial approach is to take the intersection of our s' -BSD permutation/multi-permutation codes for all $1 \leq s' \leq s$ and the redundancy is at most $s \log n + O(\log \log n)$, which is better than the best known result $(2s-1) \log n + O(1)$ from [9]. In this section, we show that we can further reduce the redundancy to $\log n + O(\log \log n)$. First we recall some terminologies in [30] used for binary $\leq s$ -burst deletion correcting codes.

Definition 5.1: For a binary vector \mathbf{x} of length n , its \mathbf{p} -indicator vector $\mathbb{1}_{\mathbf{p}}(\mathbf{x})$, where $\mathbf{p} \in \Sigma_2^m$ with $m \leq n$, is a vector of length n defined as

$$\mathbb{1}_{\mathbf{p}}(\mathbf{x})_i = \begin{cases} 1, & \text{if } \mathbf{x}_{[i, i+m-1]} = \mathbf{p}, 1 \leq i \leq n - m + 1; \\ 0, & \text{otherwise.} \end{cases}$$

Furthermore, let $n_{\mathbf{p}}(\mathbf{x})$ be the number of ones in $\mathbb{1}_{\mathbf{p}}(\mathbf{x})$ and $\alpha_{\mathbf{p}}(\mathbf{x})$ be a vector of length $n_{\mathbf{p}}(\mathbf{x}) + 1$ whose i -th entry is the distance between coordinates of the i -th and $(i + 1)$ -th 1 in the vector $(1, \mathbb{1}_{\mathbf{p}}(\mathbf{x}), 1)$.

Definition 5.2: Let $\mathbf{p} \in \Sigma_2^m$ and $\delta > m$ be a positive integer. A vector \mathbf{x} is called a (\mathbf{p}, δ) -dense vector, if each interval of length δ in \mathbf{x} contains at least one consecutive subsequence \mathbf{p} , i.e., for each $i \in [n - \delta + 1]$, there exists $j \in [i, i + \delta - m]$, such that $\mathbf{p} = (x_j, x_{j+1}, \dots, x_{j+m-1})$. Let $\mathcal{D}_{\mathbf{p}, \delta}^n$ denote all (\mathbf{p}, δ) -dense vectors in Σ_2^n .

In the rest of the paper we set $\mathbf{p} = 0^s 1^s$ and fix $\delta = s2^{2s+1} \lceil \log n \rceil$. Note that the structure of \mathbf{p} indicates that $\alpha_{\mathbf{p}}(\mathbf{x})_i \geq 2s$ for $i \geq 2$. Furthermore, $\alpha_{\mathbf{p}}(\mathbf{x})_i \leq \delta$ for any $i \in [n_{\mathbf{p}}(\mathbf{x}) + 1]$ if and only if \mathbf{x} is a (\mathbf{p}, δ) -dense vector.

Lemma 5.3 (Lemma 2, [30]): For any integer $a_1 \in \mathbb{Z}_4$ and $a_2 \in \mathbb{Z}_{2n}$, let $\mathcal{C}_{a_1, a_2}^{\text{locate}}(n, s)$ be the set

$$\left\{ \mathbf{x} \in \Sigma_2^n : \mathbf{x} \in \mathcal{D}_{\mathbf{p}, \delta}^n, n_{\mathbf{p}}(\mathbf{x}) = a_1 \pmod{4}, \right. \\ \left. \text{VT}(\alpha_{\mathbf{p}}(\mathbf{x})) = a_2 \pmod{2n} \right\}.$$

Suppose $\mathbf{x} \in \mathcal{C}_{a_1, a_2}^{\text{locate}}(n, s)$ suffers a burst deletion of length s' where $1 \leq s' \leq s$, the decoder can find an interval of length at most $\delta + s' - 1$ which contains all erroneous coordinates of \mathbf{x} .

In order to borrow Lemma 5.3 to locate all erroneous coordinates of a permutation or a multi-permutation in a bounded interval, we need a map converting permutations and multi-permutations to binary vectors which satisfies the following constraints.

- I. When a permutation (multi-permutation) suffers a burst stable deletion of length s ($s \geq 2$), the corresponding binary vector also suffers a burst deletion of length s .
- II. The permutations or multi-permutations whose corresponding binary vector is (\mathbf{p}, δ) -dense account for a large proportion of the set S_n or S_n^r (and they are referred to as (\mathbf{p}, δ) -dense permutations/multi-permutations).

The parity indicator vector defined below is a map satisfying these two constraints.

Definition 5.4: For an integer vector σ , its parity indicator vector $\mathbb{1}(\sigma)$ is defined by

$$\mathbb{1}(\sigma)_i = \begin{cases} 1, & \text{if } \sigma_i \text{ is odd;} \\ 0, & \text{if } \sigma_i \text{ is even.} \end{cases}$$

It is routine to check that when σ suffers a burst stable deletion of length s ($s \geq 2$), its parity indicator vector also suffers a burst deletion of length s . To verify Constraint II we prove the following lemma for permutations. The generalization to multi-permutations is straightforward.

Lemma 5.5: The number of (\mathbf{p}, δ) -dense permutations is at least $\frac{n!}{2}$.

Proof: Uniformly choose a random permutation $\sigma \in S_n$. Let \mathbb{E}_i be the event that $(\mathbb{1}(\sigma)_i, \mathbb{1}(\sigma)_{i+1}, \dots, \mathbb{1}(\sigma)_{i+\delta-1})$

does not contain $\mathbf{p} = 0^s 1^s$ as a consecutive subsequence, $i \in [1, n - \delta + 1]$. The probability of \mathbb{E}_i can be calculated by

$$\Pr(\mathbb{E}_i) \leq \prod_{j=0}^{\frac{\delta}{2s}-1} \Pr((\mathbb{1}(\sigma)_{[i+2sj, i+2s(j+1)-1]}) \neq \mathbf{p}) \\ = \left(1 - \frac{\binom{n/2}{s}^2 \cdot (s!)^2}{\binom{n}{2s} \cdot (2s)!} \right)^{\frac{\delta}{2s}} \\ = \left(1 - \frac{1}{2^{2s}} \cdot \frac{n(n-2) \cdots (n-2s+2)}{(n-1)(n-3) \cdots (n-2s+1)} \right)^{\frac{\delta}{2s}} \\ \leq \left(1 - \frac{1}{2^{2s}} \right)^{\frac{\delta}{2s}} \\ = \exp \left\{ \frac{\delta}{2s} \ln \left(1 - \frac{1}{2^{2s}} \right) \right\} \\ \stackrel{(*)}{\leq} \exp \left\{ -\frac{1}{2^{2s}} 2^{2s} \lceil \log n \rceil \right\} \leq \frac{1}{n^{\log e}} \leq \frac{1}{2n},$$

where $(*)$ holds since $\ln(1+x) \leq x$ for all $x \in \mathbb{R}$.

By the union bound, the probability of the event that σ is not a (\mathbf{p}, δ) -dense permutation is upper bounded by

$$\Pr(\sigma \text{ is not a } (\mathbf{p}, \delta)\text{-dense permutation}) \\ \leq \sum_{i=1}^{n-\delta+1} \Pr(\mathbb{E}_i) \leq n \cdot \frac{1}{2n} = \frac{1}{2},$$

and thus the number of (\mathbf{p}, δ) -dense permutations is at least $\frac{n!}{2}$. \blacksquare

We are now ready to present our codes against variable length burst stable deletions. Subsection V-A deals with permutations and Subsection V-B deals with multi-permutations.

A. $\leq s$ -BSD permutation codes

Given s and P , we define the PSVT codes as follows (which means the SVT version for permutations).

Construction 5.6: Let $b_1 \in \mathbb{Z}_P$, $b_2 \in \mathbb{Z}_2$, and $c_1, c_2 \in \mathbb{Z}_{2Ps}$, define the code $\text{PSVT}_{b_1, b_2, c_1, c_2}(n, s, P)$ as

$$\left\{ \sigma \in S_n : \sigma \in \mathcal{C}_{c_1, c_2}^{\text{retrieve}}(n, s, P), \right. \\ \left. \alpha(\sigma_{(s,1)}) \in \text{SVT}_{b_1, b_2}(n, P) \right\}.$$

Lemma 5.7: Suppose an arbitrary permutation σ from $\text{PSVT}_{b_1, b_2, c_1, c_2}(n, s, P)$ suffers a burst stable deletion of length s and the erroneous coordinate of $\sigma_{(s,1)}$ is within a known interval of length P . Then $\sigma_{(s,1)}$ can be recovered and the exact erroneous coordinate of $\sigma_{(s,1)}$ can be located.

Proof: Suppose $\sigma \in \text{PSVT}_{b_1, b_2, c_1, c_2}(n, s, P)$ suffers a burst stable deletion of length s and the erroneous coordinate of $\sigma_{(s,1)}$ is within a known interval of length P . Since $\sigma \in \mathcal{C}_{c_1, c_2}^{\text{retrieve}}(n, s, P)$, by Theorem 3.11 we can retrieve the missing symbol of $\sigma_{(s,1)}$. By an n -ary SVT decoder, the erroneous coordinate of $\alpha(\sigma_{(s,1)})$ lies in a consecutive run of 0's or 1's, which means that the erroneous coordinate of $\sigma_{(s,1)}$ lies in a monotone decreasing sequence or a monotone increasing sequence. In such a sequence, the coordinate of the known

missing symbol is certainly unique. Therefore, the erroneous coordinate of $\sigma_{(s,1)}$ can be exactly located. ■

We make use of the PSVT codes above for $2 \leq s' \leq s$ to finally construct our $\leq s$ -BSD permutation codes as follows.

Construction 5.8: Assume that $\frac{n}{s}$ is even and n is a multiple of $2 \prod_{s'=1}^s s' P_{s'}$ where $P_{s'} = \lceil (\delta + s' - 1)/s' \rceil$, $1 \leq s' \leq s$. Let $\mathbf{a} = (a_1, a_2)$ where $a_1 \in \mathbb{Z}_4, a_2 \in \mathbb{Z}_{2n}$. Let $\mathbf{b} = (b_1, b_2, \dots, b_{2s})$ where $b_{2s'-1} \in \mathbb{Z}_{P_{s'}}, b_{2s'} \in \mathbb{Z}_2$ for $1 \leq s' \leq s$. Let $\mathbf{c} = (c_3, c_4, \dots, c_{2s})$ where $c_{2s'-1}, c_{2s'} \in \mathbb{Z}_{2s' P_{s'}}$ for $2 \leq s' \leq s$. Let $\mathbf{d} = (d_1, d_2)$ where $d_1, d_2 \in \mathbb{Z}_{(4s)!}$. Define the code $\mathcal{C}_s^3 \triangleq \mathcal{C}_s^3(n; \mathbf{a}; \mathbf{b}, \mathbf{c}; \mathbf{d})$ as

$$\left\{ \begin{aligned} \sigma &\in S_n : \mathbb{1}(\sigma) \in \mathcal{C}_{a_1, a_2}^{\text{locate}}(n, s), \\ \alpha(\sigma) &\in \text{SVT}_{b_1, b_2}(n, P_1), \\ \sigma &\in \mathcal{C}_{d_1, d_2}^{\text{recover}}(n, 2s), \\ \sigma &\in \bigcap_{s'=2}^s \text{PSVT}_{b_{2s'-1}, b_{2s'}, c_{2s'-1}, c_{2s'}}(n, s', P_{s'}) \end{aligned} \right\}.$$

Theorem 5.9: The permutation code \mathcal{C}_s^3 is an $\leq s$ -BSD permutation code over S_n . Moreover, by choosing proper parameters there is an $\leq s$ -BSD permutation code with redundancy at most $\log n + (3s - 2) \log \log n + O(1)$.

Proof: Suppose $\sigma = (\sigma_1, \sigma_2, \dots, \sigma_n) \in \mathcal{C}_s^3(n; \mathbf{a}; \mathbf{b}, \mathbf{c}; \mathbf{d})$ suffers a burst stable deletion of length s' where $1 \leq s' \leq s$. Since its parity indicator vector $\mathbb{1}(\sigma) \in \mathcal{C}_{a_1, a_2}^{\text{locate}}(n, s)$, by Lemma 5.3 all erroneous coordinates can be located within an interval of length at most $\delta + s' - 1$. Now we view σ in an $s' \times \frac{n}{s'}$ array, and then the erroneous coordinate in the first row of such an array is located in an interval of length at most $P_{s'}$.

For $s' = 1$, the unique missing symbol of a permutation can be trivially retrieved. Since $\alpha(\sigma) \in \text{SVT}_{b_1, b_2}(n, P_1)$, we can recover σ by an n -ary SVT decoder.

For $2 \leq s' \leq s$, $\sigma \in \text{PSVT}_{b_{2s'-1}, b_{2s'}, c_{2s'-1}, c_{2s'}}(n, s', P_{s'})$. By viewing σ in an $s' \times \frac{n}{s'}$ array, in the first row $\sigma_{(s',1)}$ the erroneous coordinate is within a known interval of length $P_{s'}$. By Lemma 5.7 and Theorem 3.11, $\sigma_{(s',1)}$ can be recovered and the erroneous coordinate of $\sigma_{(s',1)}$ is exactly located, and thus all erroneous coordinates of σ are located in an interval of length $2s'$. Then we view σ in an $2s \times \frac{n}{2s}$ array and all erroneous coordinates can be located in two adjacent columns in this array. Finally, since $\sigma \in \mathcal{C}_{d_1, d_2}^{\text{recover}}(n, 2s)$, the correction of the two columns can be done in a similar way as Lemma 4.8.

By the pigeon-hole principal, there exist parameters $a_1 \in \mathbb{Z}_4, a_2 \in \mathbb{Z}_{2n}, b_{2s'-1} \in \mathbb{Z}_{P_{s'}}, b_{2s'} \in \mathbb{Z}_2$ for $1 \leq s' \leq s$, $c_{2s'-1}, c_{2s'} \in \mathbb{Z}_{2s' P_{s'}}$ for $2 \leq s' \leq s$, and $d_1, d_2 \in \mathbb{Z}_{(4s)!}$, such that the redundancy of \mathcal{C}_s^3 is at most

$$\begin{aligned} &\log 2 + \log 4 + \log 2n + \sum_{s'=1}^s (\log P_{s'} + \log 2) \\ &\quad + \sum_{s'=2}^s 2 \log 2s' P_{s'} + 2 \log(4s)! \\ &= \log n + (3s - 2) \log \log n + O(1), \end{aligned}$$

where the first term is the 1 bit redundancy due to the size of (p, δ) -dense permutations (Lemma 5.5). ■

B. $\leq s$ -BSD multi-permutation codes

Due to the existence of repeated entries in a multi-permutation, we need a little modification as we do in the previous section. Given s and P , we define the MPSVT codes as follows (which means the SVT version for multi-permutations).

Construction 5.10: Let $\mathbf{r} = (r_1, r_2, \dots, r_w)$ be a multiplicity vector. Let $b_1 \in \mathbb{Z}_P, b_2 \in \mathbb{Z}_2$, and $c_1, c_2 \in \mathbb{Z}_{2P_s}$, define the code $\text{MPSVT}_{b_1, b_2, c_1, c_2}(n, s, P)$ as

$$\left\{ \begin{aligned} \sigma &\in S_n^{\mathbf{r}} : \sigma \in \mathcal{C}_{c_1, c_2}^{\text{retrieve}}(n, s, P), \\ \alpha(\sigma_{(s,1)}) &\in \text{SVT}_{b_1, b_2}(n, P) \end{aligned} \right\}.$$

Lemma 5.11: Suppose an arbitrary multi-permutation $\sigma \in \text{MPSVT}_{b_1, b_2, c_1, c_2}(n, s, P)$ suffers a burst stable deletion of length s and the erroneous coordinate of $\sigma_{(s,1)}$ is within a known interval of length P . Then $\sigma_{(s,1)}$ can be recovered and the exact erroneous coordinate of $\sigma_{(s,1)}$ can be located within an interval of length r in the worst case, where $r = \max_{1 \leq i \leq w} r_i$.

Lemma 5.11 can be proved analogously as Lemma 5.7, with the only difference in the final step due to the existence of repeated entries. We make use of the MPSVT codes above for $2 \leq s' \leq s$ to finally construct our $\leq s$ -BSD multi-permutation codes as follows, assuming the multiplicity vector \mathbf{r} is regular.

Construction 5.12: Assume $n = rw = s(r+1)t$ where t is even. Further we assume that n is a multiple of $2 \prod_{s'=1}^s s' P_{s'}$ where $P_{s'} = \lceil (\delta + s' - 1)/s' \rceil$, $1 \leq s' \leq s$. Let $\mathbf{a} = (a_1, a_2)$ where $a_1 \in \mathbb{Z}_4, a_2 \in \mathbb{Z}_{2n}$. Let $\mathbf{b} = (b_1, b_2, \dots, b_{2s})$ where $b_{2s'-1} \in \mathbb{Z}_{P_{s'}}, b_{2s'} \in \mathbb{Z}_2$ for $1 \leq s' \leq s$. Let $\mathbf{c} = (c_3, c_4, \dots, c_{2s})$ where $c_{2s'-1}, c_{2s'} \in \mathbb{Z}_{2s' P_{s'}}$ for $2 \leq s' \leq s$. Let $\mathbf{d} = (d_1, d_2)$ where $d_1, d_2 \in \mathbb{Z}_{(2s(r+1))!}$. Define the code $\mathcal{C}_s^4 \triangleq \mathcal{C}_s^4(n; \mathbf{a}; \mathbf{b}, \mathbf{c}; \mathbf{d})$ as

$$\left\{ \begin{aligned} \sigma &\in S_n^{\mathbf{r}} : \mathbb{1}(\sigma) \in \mathcal{C}_{a_1, a_2}^{\text{locate}}(n, s), \\ \alpha(\sigma) &\in \text{SVT}_{b_1, b_2}(n, P_1), \\ \sigma &\in \mathcal{C}_{d_1, d_2}^{\text{recover}}(n, s(r+1)), \\ \sigma &\in \bigcap_{s'=2}^s \text{MPSVT}_{b_{2s'-1}, b_{2s'}, c_{2s'-1}, c_{2s'}}(n, s', P_{s'}) \end{aligned} \right\}.$$

Theorem 5.13: The code \mathcal{C}_s^4 we constructed in Construction 5.12 is an $\leq s$ -BSD multi-permutation code over $S_n^{\mathbf{r}}$ (\mathbf{r} is a regular multiplicity vector). Moreover, by choosing proper parameters there is an $\leq s$ -BSD multi-permutation code over $S_n^{\mathbf{r}}$ with redundancy at most $\log n + (3s - 2) \log \log n + O(1)$.

Proof: The proof works in the same way as the proof of Theorem 5.9 until we recover $\sigma_{(s',1)}$. However, due to repeated entries in multi-permutations, here we can only locate the erroneous coordinate of $\sigma_{(s',1)}$ in an interval of length r , in the worst case. Thus, all erroneous coordinates of σ are located in an interval of length $s'(r+1)$. By viewing σ in an array of size $s(r+1) \times \frac{n}{s(r+1)}$, all the erroneous coordinates are within two adjacent columns of this array. Finally, since $\sigma \in \mathcal{C}_{d_1, d_2}^{\text{recover}}(n, s(r+1))$, the correction of the two columns can be done in a similar way as Lemma 4.8.

The code redundancy can be calculated by the pigeon-hole principle immediately. ■

To construct codes for irregular multi-permutations, simply choose $r_{\max} = \max_{i \in [w]} r_i$ and apply the above construction.

Corollary 5.14: There exists an $\leq s$ -BSD multi-permutation code in S_n^r (r could be irregular) with redundancy at most $\log n + (3s - 2) \log \log n + O(1)$.

VI. A LINEAR-TIME ENCODER FOR SINGLE STABLE DELETION CORRECTING PERMUTATION CODES

Previous sections consist of the stable deletion correcting permutation or multi-permutation codes, together with their decoding algorithms. In this section, we consider the problem of designing an efficient encoder that encodes a user message, denoted by an arbitrary permutation $\pi \in S_k$, into a permutation lying in a stable deletion correcting permutation code $\mathcal{C} \subseteq S_n$.

It should be noted that designing efficient encoders in deletion error models is quite a challenging task. For the binary case, the encoder proposed by Abdel-Ghaffar and Ferreira [1] with almost optimal redundancy for the binary VT code did not appear until 1998. For the q -ary case, so far there is no efficient encoder for the q -ary VT code $VT_{a,b}(n, q)$ matching the optimal redundancy, and the existential best known encoder proposed by Abroshan et al. [2] has redundancy $(\log q + 1)\lceil \log n \rceil + 2(\log q - 1)$. A very recent result [33] introduces a variant of q -ary VT code with more efficient encoder.

To the best of our knowledge, so far there is no research on efficient encoders for deletion correcting permutation codes, even for only a single deletion. Recall that the code $\mathcal{C}_a(n) = \{\sigma \in S_n : \alpha(\sigma) \in VT_a(n-1)\}$, where $0 \leq a < n$, is a single stable deletion correcting permutation code given by Levenshtein [28]. In this section, we give a linear-time encoder which encodes an arbitrary permutation $\pi \in S_{n-1}$ into a permutation in $\mathcal{C}_a(n)$ and thus the encoder has the optimal redundancy $\log n$. The key idea is to show that we can simply find an appropriate coordinate to insert the symbol n and guarantee that the resultant permutation indeed belongs to $\mathcal{C}_a(n)$. The encoder and the proof of its correctness are as follows.

Theorem 6.1: Algorithm 1 is correct and runs in $O(n)$ time.

Proof: For any $\pi \in S_{n-1}$, let $\Delta = (a - \sum_{i=1}^{n-2} i\alpha(\pi_i)) \pmod n$ and $\omega = \sum_{i=1}^{n-2} \alpha(\pi_i)$. Then we have $0 \leq \Delta \leq n-1$ and $0 \leq \omega \leq n-2$.

If $0 \leq \Delta \leq \omega - 1$, let j be the coordinate of the $(\omega - \Delta)$ -th 1 in $\alpha(\pi)$. Then, $\alpha(\pi_j) = 1$, i.e., $\pi_j \leq \pi_{j+1}$, and we insert n between π_j and π_{j+1} , i.e., $\sigma = (\pi_1, \dots, \pi_j, n, \pi_{j+1}, \dots, \pi_{n-1})$. We obtain

$$\begin{aligned} \alpha(\sigma) &= (\alpha(\pi_1), \dots, \alpha(\pi_{j-1}), 1, 0, \alpha(\pi_{j+1}), \dots, \alpha(\pi_{n-2})) \\ &= (\alpha(\pi_1), \dots, \alpha(\pi_j), 0, \alpha(\pi_{j+1}), \dots, \alpha(\pi_{n-2})), \end{aligned}$$

that is, $\alpha(\sigma)$ is obtained by inserting a symbol 0 at the j -th

Algorithm 1: Single stable deletion correcting permutation codes encoder

Input: $\pi \in S_{n-1}$ and $0 \leq a < n$
Output: $\sigma = (\pi_1, \dots, \pi_j, n, \pi_{j+1}, \dots, \pi_{n-1}) \in \mathcal{C}_a(n)$
1 Initialization: Let $\Delta = (a - \sum_{i=1}^{n-2} i\alpha(\pi_i)) \pmod n$
and $\omega = \sum_{i=1}^{n-2} \alpha(\pi_i)$
2 if $0 \leq \Delta \leq \omega - 1$ **then**
3 | Let j be the coordinate of the $(\omega - \Delta)$ -th 1 in
| $\alpha(\pi)$
4 end
5 if $\Delta = \omega$ **then**
6 | Let $j = 0$
7 end
8 if $\omega < \Delta < n - 1$ **then**
9 | Let j be the coordinate of $(\Delta - \omega)$ -th 0 in $\alpha(\pi)$
10 end
11 if $\Delta = n - 1$ **then**
12 | Let $j = n - 1$
13 end

coordinate in $\alpha(\pi)$. Now, we have

$$\begin{aligned} \sum_{i=1}^{n-1} i\alpha(\sigma_i) &= \sum_{i=1}^j i\alpha(\pi_i) + \sum_{i=j+2}^{n-1} i\alpha(\pi_{i-1}) \\ &= \sum_{i=1}^{n-2} i\alpha(\pi_i) + \sum_{i=j+1}^{n-2} \alpha(\pi_i) \\ &= (a - \Delta) + \omega - (\omega - \Delta) \\ &= a \pmod n, \end{aligned}$$

and thus indeed we have $\sigma \in \mathcal{C}_a(n)$.

If $\Delta = \omega$, we insert n at the very beginning of π , that is, $\sigma = (n, \pi_1, \dots, \pi_{n-1})$, and we obtain $\alpha(\sigma) = (0, \alpha(\pi_1), \dots, \alpha(\pi_{n-2}))$, that is, $\alpha(\sigma)$ is obtained by inserting a symbol 0 at the very beginning of $\alpha(\pi)$. Now, we have

$$\begin{aligned} \sum_{i=1}^{n-1} i\alpha(\sigma_i) &= \sum_{i=2}^{n-1} i\alpha(\pi_{i-1}) \\ &= \sum_{i=1}^{n-2} i\alpha(\pi_i) + \sum_{i=1}^{n-2} \alpha(\pi_i) \\ &= (a - \Delta) + \Delta \\ &= a \pmod n, \end{aligned}$$

and thus indeed we have $\sigma \in \mathcal{C}_a(n)$.

If $\omega < \Delta < n - 1$, let j be the coordinate of $(\Delta - \omega)$ -th 0 in $\alpha(\pi)$. Note that $\alpha(\sigma)$ has $n - 2 - \omega$ zeros and thus such a coordinate must exist. Then, $\alpha(\pi_j) = 0$, i.e., $\pi_j > \pi_{j+1}$, and we insert n between π_j and π_{j+1} , i.e., $\sigma = (\pi_1, \dots, \pi_j, n, \pi_{j+1}, \dots, \pi_{n-1})$. We obtain

$$\begin{aligned} \alpha(\sigma) &= (\alpha(\pi_1), \dots, \alpha(\pi_{j-1}), 1, 0, \alpha(\pi_{j+1}), \dots, \alpha(\pi_{n-2})) \\ &= (\alpha(\pi_1), \dots, \alpha(\pi_{j-1}), 1, \alpha(\pi_j), \alpha(\pi_{j+1}), \dots, \alpha(\pi_{n-2})), \end{aligned}$$

that is, $\alpha(\sigma)$ is obtained by inserting a symbol 1 at the $(j-1)$ -th coordinate in $\alpha(\pi)$. Now, we have

$$\begin{aligned} \sum_{i=1}^{n-1} i\alpha(\sigma_i) &= \sum_{i=1}^{j-1} i\alpha(\pi_i) + j + \sum_{i=j+1}^{n-1} i\alpha(\pi_{i-1}) \\ &= \sum_{i=1}^{n-2} i\alpha(\pi_i) + j + \sum_{i=j}^{n-2} \alpha(\pi_i) \\ &= (a - \Delta) + j + (n - 2 - j + 1) \\ &\quad - (n - 2 - \omega - (\Delta - \omega - 1)) \\ &= a \pmod{n}, \end{aligned}$$

and thus indeed we have $\sigma \in \mathcal{C}_a(n)$.

Finally, if $\Delta = n - 1$, we insert n at the very end of π , that is, $\sigma = (\pi_1, \dots, \pi_{n-1}, n)$. We obtain $\alpha(\sigma) = (\alpha(\pi_1), \dots, \alpha(\pi_{n-2}), 1)$, that is, $\alpha(\sigma)$ is obtained by inserting a symbol 1 at the very end of $\alpha(\pi)$. Now, we have

$$\begin{aligned} \sum_{i=1}^{n-1} i\alpha(\sigma_i) &= \sum_{i=1}^{n-2} i\alpha(\pi_i) + n - 1 \\ &= (a - \Delta) + n - 1 \\ &= a \pmod{n}, \end{aligned}$$

and thus indeed we have $\sigma \in \mathcal{C}_a(n)$.

To sum up, for each of the four cases, we encode $\pi \in S_{n-1}$ as $\sigma \in \mathcal{C}_a(n)$. Furthermore, it is straightforward to check that we can calculate Δ and ω and find j in $O(n)$ time. ■

Example 6.2: Let $n = 10$ and $a = 0$.

- If $\pi = (2, 1, 4, 3, 6, 5, 8, 7, 9)$, then we have $\alpha(\pi) = (0, 1, 0, 1, 0, 1, 0, 1)$. Now, let $\Delta = (0 - \sum_{i=1}^8 i\alpha(\pi_i)) \pmod{10} = 0$ and $\omega = \sum_{i=1}^8 \alpha(\pi_i) = 4$. Then, $0 \leq \Delta \leq \omega - 1$, the coordinate of 4-th 1 ($\omega - \Delta = 4$) in $\alpha(\pi)$ is 8, and we let $\sigma = (2, 1, 4, 3, 6, 5, 8, 7, 10, 9)$. Thus, we obtain $\alpha(\sigma) = (0, 1, 0, 1, 0, 1, 0, 1, 0)$ and $\sum_{i=1}^9 i\alpha(\sigma_i) = 0 \pmod{10}$.
- If $\pi = (1, 2, 4, 3, 9, 8, 7, 6, 5)$, then we have $\alpha(\pi) = (1, 1, 0, 1, 0, 0, 0, 0)$. Now, let $\Delta = (0 - \sum_{i=1}^8 i\alpha(\pi_i)) \pmod{10} = 3$ and $\omega = \sum_{i=1}^8 \alpha(\pi_i) = 3$. Then $\Delta = \omega$ and we let $\sigma = (10, 1, 2, 4, 3, 9, 8, 7, 6, 5)$. Thus, we obtain that $\alpha(\sigma) = (0, 1, 1, 0, 1, 0, 0, 0, 0, 0)$ and $\sum_{i=1}^9 i\alpha(\sigma_i) = 0 \pmod{10}$.
- If $\pi = (3, 1, 2, 9, 8, 7, 6, 5, 4)$, then we have $\alpha(\pi) = (0, 1, 1, 0, 0, 0, 0, 0)$. Now, let $\Delta = (0 - \sum_{i=1}^8 i\alpha(\pi_i)) \pmod{10} = 5$ and $\omega = \sum_{i=1}^8 \alpha(\pi_i) = 2$. Then $\omega < \Delta < n - 1$, the coordinate of 3-th 0 ($\Delta - \omega = 3$) in $\alpha(\pi)$ is 5, and we let $\sigma = (3, 1, 2, 9, 8, 10, 7, 6, 5, 4)$. Now, we have $\alpha(\sigma) = (0, 1, 1, 0, 1, 0, 0, 0, 0, 0)$ and $\sum_{i=1}^9 i\alpha(\sigma_i) = 0 \pmod{10}$.
- If $\pi = (1, 9, 8, 7, 6, 5, 4, 3, 2)$, then we have $\alpha(\pi) = (1, 0, 0, 0, 0, 0, 0, 0)$. Now, let $\Delta = (0 - \sum_{i=1}^8 i\alpha(\pi_i)) \pmod{10} = 9 = n - 1$. Then we let the permutation $\sigma = (1, 9, 8, 7, 6, 5, 4, 3, 2, 10)$. Thus, we obtain that $\alpha(\sigma) = (1, 0, 0, 0, 0, 0, 0, 0, 0, 1)$ and $\sum_{i=1}^9 i\alpha(\sigma_i) = 0 \pmod{10}$.

Moreover, our encoding algorithm clearly suggests a rather simple decoding algorithm, presented in Algorithm 2.

Algorithm 2: Single stable deletion correcting permutation code decoder

Input: $\sigma' \in \mathcal{B}_1(\sigma)$ where $\sigma \in \mathcal{C}_a(n)$
Output: $\pi \in S_{n-1}$
1 Step 1: Recover σ from σ'
2 Step 2: Find the symbol n in σ and remove it

VII. CONCLUSION

Motivated by applications in flash memories, in this paper we consider permutation codes and multi-permutation codes against burst stable deletions. Our main improvement relies on a different approach to retrieve the missing symbol on the first row of the array representation of a permutation or a multi-permutation, with less redundancy. The gap between the redundancy of our codes and the theoretic bound is only of order $\log \log n$. The more difficult model of burst unstable deletions for permutations and multi-permutations are considered for future research. Another interesting problem is to investigate the efficient encoder for burst stable deletion correcting permutation and multi-permutation codes.

REFERENCES

- [1] K. A. S. Abdel-Ghaffar and H. C. Ferreira, "Systematic encoding of the Varshamov-Tenengol'ts codes and the Constantin-Rao codes," *IEEE Trans. Inf. Theory*, vol. 44, no. 1, pp. 340-345, Jan. 1998.
- [2] M. Abroshan, R. Venkataramanan, and A. G. I. Fabregas, "Efficient Systematic Encoding of Non-binary VT Codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Vail, CO, USA, Jun. 2018, pp. 91-95.
- [3] 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, Aug. 2010.
- [4] J. Brakensiek, V. Guruswami, and S. Zbarsky, "Efficient low-redundancy codes for correcting multiple deletions," *IEEE Trans. Inf. Theory*, vol. 64, no. 5, pp. 3403-3410, May 2018.
- [5] R. Bitar, S. K. Hanna, N. Polyanskii, and I. Vorobyev, "Optimal codes correcting localized deletions," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Melbourne, Australia, July. 2021, pp. 1991-1996.
- [6] 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 (ISIT)*, Hong Kong, China, Jun. 2015, pp. 1741-1745.
- [7] Y. M. Chee, S. Ling, 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 (ISIT)*, Aachen, Germany, Jun. 2017, pp. 2688-2692.
- [8] Y. M. Chee, H. M. Kiah, A. Vardy, V. K. Vu, and E. Yaakobi, "Coding for racetrack memories," *IEEE Trans. Inf. Theory*, vol. 64, no. 11, pp. 7094-7112, Nov. 2018.
- [9] Y. M. Chee, S. Ling, T. T. Nguyen, V. K. Vu, H. Wei, and X. Zhang, "Burst-deletion-correcting codes for permutations and multipermutations," *IEEE Trans. Inf. Theory*, vol. 66, no. 2, pp. 957-969, Feb. 2020.
- [10] F. Farnoud, V. Skachek, and O. Milenkovic, "Error-correction in flash memories via codes in the ulam metric," *IEEE Trans. Inf. Theory*, vol. 59, no. 5, pp. 3003-3020, May 2013.
- [11] 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 (ISIT)*, Cambridge, MA, USA, Jul. 2012, pp. 990-994.
- [12] 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.
- [13] R. Gabrys, E. Yaakobi, and O. Milenkovic, "Codes in the Damerau distance for deletion and adjacent transposition correction," *IEEE Trans. Inf. Theory*, vol. 64, no. 4, pp. 2550-2570, Apr. 2018.
- [14] R. Gabrys and F. Sala, "Codes correcting two deletions," *IEEE Trans. Inf. Theory*, vol. 65, no. 2, pp. 965-974, Feb. 2019.
- [15] V. Guruswami and J. Håstad, "Explicit two-deletion codes with redundancy matching the existential bound," *IEEE Trans. Inf. Theory*, vol. 67, no. 10, pp. 6384-6394, Oct. 2021.

- [16] F. F. Hassanzadeh and O. Milenkovic, "Multipermutation codes in the ulam metric for nonvolatile memories," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 5, pp. 919-932, May 2014.
- [17] B. Haeupler and A. Shahrabi, "Synchronization strings: Codes for insertions and deletions approaching the singleton bound," *J. ACM*, vol. 68, no. 5, pp. 1-39, Sep. 2021.
- [18] 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.
- [19] H. Han, J. Mu, Y.-C. He, X. Jiao, and W. Ma, "Rate-improved permutation codes for correcting a single burst of deletions," *IEEE Commun. Lett.*, vol. 25, no. 1, pp. 45-93, Jan. 2021.
- [20] S. K. Hanna and S. El Rouayheb, "Guess & check codes for deletions, insertions, and synchronization," *IEEE Trans. Inf. Theory*, vol. 65, no. 1, pp. 3-15, Jan. 2019.
- [21] S. K. Hanna and S. El Rouayheb, "Codes for Correcting Localized Deletions," *IEEE Trans. Inf. Theory*, vol. 67, no. 4, pp. 2206-2216, Apr. 2021.
- [22] M. Horowitz and T. Etzion, "Local rank modulation for flash memories," *IEEE Trans. Inf. Theory*, vol. 65, no. 3, pp. 1705-1713, Nov. 2019.
- [23] 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.
- [24] 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.
- [25] 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.
- [26] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet Phys. Dokl.*, vol. 10, no. 8, pp. 707-710, Feb. 1966.
- [27] V. I. Levenshtein, "Asymptotically optimum binary codes with correction for losses of one or two adjacent bits," *Syst. Theory Res.*, vol. 19, pp. 298-304, 1970.
- [28] V. I. Levenshtein, "On perfect codes in deletion and insertion metric," *Discrete. Math. Appl.*, vol. 2, no. 3, pp. 241-258, 1992.
- [29] M. Levy and E. Yaakobi, "Mutually uncorrelated codes for DNA storage," *IEEE Trans. Inf. Theory*, vol. 65, no. 6, pp. 3671-3691, Jun. 2019.
- [30] A. Lenz and N. Polyanskii, "Optimal codes correcting a burst of deletions of variable length," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Los Angeles, CA, USA, Jun. 2020, pp. 757-762.
- [31] M. Mitzenmacher, "A survey of results for deletion channels and related synchronization channels," *Probability Surveys*, vol. 6, pp. 1-33, 2009.
- [32] H. Mercier, V. Bhargava, and V. Tarokh, "A survey of error-correcting codes for channels with symbol synchronization errors," *IEEE Commun. Surv. Tutorials*, vol. 12, no. 1, pp. 87-96, 2010.
- [33] T. T. Nguyen, K. Cai, and P. H. Siegel, "Every Bit Counts: A New Version of Non-binary VT Codes with More Efficient Encoder," 2022, [arXiv:2212.10721](https://arxiv.org/abs/2212.10721).
- [34] K. Prall, "Scaling non-volatile memory below 30 nm," in *Proc. 22nd IRRR Non-Volatile Semiconductor Memory Work. (NVSMW)*, Monterey, CA, USA, Aug. 2007, pp. 5-10.
- [35] F. Sala, R. Gabrys, and L. Dolecek, "Dynamic threshold schemes for multi-level non-volatile memories," *IEEE Trans. Commun.*, vol. 61, no. 7, pp. 2624-2634, Jul. 2013.
- [36] F. Sala, R. Gabrys, and L. Dolecek, "Deletions in multipermutations," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Honolulu, HI, USA, Jun. 2014, pp. 2769-2773.
- [37] 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.
- [38] C. Schoeny, F. Sala, and L. Dolecek, "Novel combinatorial coding results for dna sequencing and data storage," in *2017 51st Asilomar Conference on Signals, Systems, and Computers (ACSSC)*, Pacific Grove, CA, USA, Oct. 2017, pp. 511-515.
- [39] T. Saeki and T. Nozaki, "An improvement of non-binary code correcting single b-burst of insertions or deletions," in *Proc. IEEE Int. Symp. Inf. Theory Appl. (ISITA)*, Singapore, Oct. 2018, pp. 6-10.
- [40] J. Sima, N. Retanel, and J. Bruck, "Two deletion correcting codes from indicator vectors," *IEEE Trans. Inf. Theory*, vol. 66, no. 4, pp. 2375-2391, Apr. 2020.
- [41] J. Sima, R. Gabrys, and J. Bruck, "Optimal systematic t-deletion correcting codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Los Angeles, CA, USA, Jun. 2020, pp. 769-774.
- [42] J. Sima, R. Gabrys, and J. Bruck, "Optimal codes for the q-ary deletion channel," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Los Angeles, CA, USA, Jun. 2020, pp. 740-745.
- [43] J. Sima and J. Bruck, "On optimal k-Deletion correcting codes," *IEEE Trans. Inf. Theory*, vol. 67, no. 6, pp. 3360-3375, Jun. 2021.
- [44] G. M. Tenengol'ts, "Nonbinary codes, correcting single deletion or insertion," *IEEE Trans. Inf. Theory*, vol. 30, no. 5, pp. 766-769, Sep. 1984.
- [45] R. R. Varshamov and G. M. Tenengol'ts, "A code that corrects single unsymmetric errors," *Avtomatika Telemekhanika*, vol. 26, no. 2, pp. 288-292, 1965.
- [46] S. H. T. Yazdi, H. M. Kiah, E. Garcia-Ruiz, J. Ma, H. Zhao, and O. Milenkovic, "DNA-based storage: Trends and methods," *IEEE Trans. Molecular, Biological, Multi-Scale Commun.*, vol. 1, no. 3, pp. 230-248, Sep. 2015.
- [47] H. Zhou, M. Schwartz, 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.