EasyChair Preprint

№ 8958

# Permutation Codes for Correcting a Burst of at Most t Deletions

Shuche Wang, Yuanyuan Tang, Ryan Gabrys and Farzad Farnoud



October 3, 2022

# Permutation Codes for Correcting a Burst of at Most $t$ Deletions

Shuche Wang*, Yuanyuan Tang†, Ryan Gabrys‡, and Farzad Farnoud†
* Institute of Operations Research and Analytics, National University of Singapore, shuche.wang@u.nus.edu
† Electrical & Computer Engineering, University of Virginia, {yt5tz,farzad}@virginia.edu
†Calit2, University of California-San Diego, U.S.A., rgabrys@eng.ucsd.edu

***Abstract*—Codes over permutations have received significant attention due to their applications in memory devices and powerline transmission systems. In this paper, we construct a family of permutation codes that are capable of correcting a burst of at most $t$ deletions with $\log n + \mathcal{O}(\log\log n)$ bits of redundancy where $t$ is a constant, which significantly improves upon the state-of-the-art construction with redundancy $2t\log n$ bits.**

## I. Introduction

Permutation codes were first proposed by Slepian [1] for data transmission in the presence of additive Gaussian noise, and have been studied in the context of powerline transmission systems to combat impulse noise [2], [3] and in the design of block ciphers [4]. More recently, permutation codes have garnered attention due to their applications in flash memories [5]–[9].

Flash memories have evolved into a sophisticated technology for nonvolatile data storage because of their advantages in terms of speed, power consumption, and reliability. To avoid the challenge of precisely programming each cell in a flash memory device to its designated level, Jiang et al. [5] proposed a rank modulation approach to express information using permutations. To overcome deletions/erasures that may arise under this setup, Gabrys et al. [10] classified deletions into two categories: *symbol-invariant deletions (SID)* and *permutation-invariant deletions (PID)*; they proposed permutation codes for correcting a single deletion under both models. Sala et al. [11] extended this work to the multipermutation setup.

In this paper, we focus on permutation codes for correcting a burst of consecutive deletions under the SID model, where deleting some symbols does not affect the values of others. Although there have been several works on binary and non-binary codes for correcting a burst of deletions [12]–[14], permutation codes for these type of errors are not as well-studied. Han et al. [15]–[17] constructed a series of permutation and multipermutation codes for correcting a burst of deletions using an interleaving technique. However, their constructions requires linear redundancy in terms of the code length. Chee et al. in [18] proposed two permutation codes for correcting a burst of exactly $t$ and at most $t$ deletions with redundancy $2\log n$ and $2t\log n$, respectively. The idea behind their code construction for correcting a burst of exactly $t$ deletions is outlined below:

- The permutation is interleaved into $t$ rows and interpreted as a $t \times \frac{n}{t}$ array.
- A Tenegolts' code, which is capable of correcting a single deletion, is applied in the first row. This code is used to correct the deletion in the first row as well as to determine the location of the burst of deletions.
- Ranking constraints are placed on each pair of adjacent columns in the remaining rows to correct the remaining errors.

For correcting a burst of at most $t$ deletions, the approach proposed by Chee et al. [18] is similar except that it requires a set of $t$ related constraints (to correct every possible burst length) resulting in a construction which requires roughly $2t\log n$ bits of redundancy.

In this paper, we construct a family of permutation codes that are capable of correcting a burst of at most $t$ deletions with redundancy $\log n + \mathcal{O}(\log\log n)$ bits. Unlike [18], we do not rely on a matrix representation of our codewords but instead introduce a simple binary map (which is formed based on the values of the symbols) to first determine approximately where the burst of deletions occurs. Afterwards, we introduce a novel secondary mapping based upon the rankings of (overlapping) groups of consecutive symbols in a given permutation to recover the exact positions and ordering of the deleted symbols.

## II. Notation and Preliminaries

We now describe the notation used throughout this paper. $\Sigma_q$ denotes a finite alphabet of size $q$ and $\Sigma_q^n$ represents the set of all sequences of length $n$ over $\Sigma_q$. Without loss of generality, we assume $\Sigma_q = \{0, 1, \ldots, q-1\}$. All logarithms in this paper are to the base 2.

We write sequences with bold letters, such as $\boldsymbol{u}$ and their elements with plain letters, e.g., $\boldsymbol{u} = u_1 \cdots u_n$ for $\boldsymbol{u} \in \Sigma_q^n$. For functions, if the output is a sequence, we also write them with bold letters, such as $\boldsymbol{\phi}(\boldsymbol{u})$. The $i$th position in $\boldsymbol{\phi}(\boldsymbol{u})$ is denoted $\phi(\boldsymbol{u})_i$. We typically use $\boldsymbol{u}$ for non-binary and $\boldsymbol{x}$ for binary sequences. The length of the sequence $\boldsymbol{x}$ is denoted $|\boldsymbol{x}|$ and $\boldsymbol{x}_{[i,j]}$ denotes the substring beginning at index $i$ and ending at index $j$, inclusive.

For ease of notation, we will denote the set $\{0, 1, \ldots, m-1\}$ as $[[m]]$ and the set $\{1, 2, \ldots, m\}$ as $[m]$. Let $n$ be a positive integer and $\mathcal{S}_n$ be the set of all permutations on the set $[n]$. Denote $\boldsymbol{\pi} = (\pi_1, \pi_2, \ldots, \pi_n) \in \mathcal{S}_n$ as a permutation with

length $n$. A *burst of at most $t$ deletions* deletes at most $t$ consecutive symbols from the permutation $\boldsymbol{\pi}$, leading to $\boldsymbol{\pi}' = (\pi_1, \pi_2, \ldots, \pi_i, \pi_{i+t'+1}, \ldots, \pi_n)$, where $t' \leq t$. Our goal in this paper will be to design codes $\mathcal{P}_{\leq t}(n)$ over $\mathcal{S}_n$ where for any $\boldsymbol{\pi} \in \mathcal{P}_{\leq t}(n)$, we can recover $\boldsymbol{\pi}$ from $\boldsymbol{\pi}'$, provided that $\boldsymbol{\pi}'$ is the result of a burst of at most $t$ deletions occurring in $\boldsymbol{\pi}$.

For a sequence of positive integers $\boldsymbol{u}$ of length $n$, the permutation projection of $\boldsymbol{u}$, denoted by $\mathbf{Prj}(\boldsymbol{u})$, is a permutation in $\mathcal{S}_n$ where:

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

**Example 1.** Let $\boldsymbol{u} = (3, 8, 4, 1, 5)$. Then $\mathbf{Prj}(\boldsymbol{u}) = (2, 5, 3, 1, 4)$. △

Let the function $\mu : \mathcal{S}_n \to [1, n!]$ be a bijection such that $\mu(\boldsymbol{\pi})$ is the lexicographic rank of $\boldsymbol{\pi}$ in $\mathcal{S}_n$. For a sequence $\boldsymbol{u}$ of length $n$, we define the permutation rank of $\boldsymbol{u}$ as $r(\boldsymbol{u}) = \mu(\mathbf{Prj}(\boldsymbol{u})) \in [1, n!]$. Also, we define the corresponding overlapping ranking sequence for a permutation $\boldsymbol{\pi}$ of length $n$ as $\boldsymbol{p}_{t+1}(\boldsymbol{\pi}) = (p_1, p_2, \ldots, p_i, \ldots, p_{n-t}) \in \Sigma_{(t+1)!}^{n-t}$, where $p_i = r(\pi_i, \pi_{i+1}, \ldots, \pi_{i+t})$.

**Example 2.** For permutations of length 3, we have $\mu(\boldsymbol{\pi}_1 = (1, 2, 3)) = 1$, $\mu(\boldsymbol{\pi}_2 = (1, 3, 2)) = 2$, $\mu(\boldsymbol{\pi}_3 = (2, 1, 3)) = 3$, $\mu(\boldsymbol{\pi}_4 = (2, 3, 1)) = 4$, $\mu(\boldsymbol{\pi}_5 = (3, 1, 2)) = 5$ and $\mu(\boldsymbol{\pi}_6 = (3, 2, 1)) = 6$. △

**Example 3.** Suppose $\boldsymbol{\pi} = (6, 4, 2, 1, 5, 3)$ and $t = 2$, then the corresponding overlapping ranking sequence $\boldsymbol{p}_3(\boldsymbol{\pi}) = (6, 6, 3, 2)$. △

The size of a permutation code $\mathcal{C} \subseteq \mathcal{S}_n$ is denoted $|\mathcal{C}|$ and its redundancy is defined as $\log(n!/|\mathcal{C}|)$.

**Proposition 1.** *(Chee et al. [18]) Let $n > t$ be positive integers, the maximum size of a $t$-burst permutation code over $\mathcal{S}_n$ is at most $n!/(t!(n - t + 1))$.*

Hence, the lower bound of the minimal redundancy of the permutation code for correcting at most $t$ deletions is $\log n + \mathcal{O}(1)$.

We now introduce several codes, which are similar to ones from [14] that will be of use in the paper for locating the deletions in an $\mathcal{O}(\log n)$ interval in Section III. Denote the indicator vector of the pattern $\boldsymbol{w}$ as $\mathbb{1}_{\boldsymbol{w}}$:

$$\mathbb{1}_{\boldsymbol{w}}(\boldsymbol{x})_i = \begin{cases} 1, & \text{if } \boldsymbol{x}_{[i,i+|\boldsymbol{w}|-1]} = \boldsymbol{w} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

Let $n_{\boldsymbol{w}}$ be the number of ones in $\mathbb{1}_{\boldsymbol{w}}$ and define $\boldsymbol{\alpha}_{\boldsymbol{w}}(\boldsymbol{x})$ to be a vector of length $n_{\boldsymbol{w}} + 1$ whose $i$-th entry is the distance between positions of the $i$-th and $(i + 1)$-th 1 in the string $(1, \mathbb{1}_{\boldsymbol{w}}, 1)$.

Fix the pattern $\boldsymbol{w} = \mathbf{0}^t\mathbf{1}^t$ and $\delta = t2^{2t+2}\lceil \log n \rceil$. We now introduce the set of $(\boldsymbol{w}, \delta)$-dense strings:

$$\mathcal{D}_{\boldsymbol{w},\delta}(n) = \{\boldsymbol{x} \in \Sigma_2^n : \boldsymbol{\alpha}_{\boldsymbol{w}}(\boldsymbol{x})_i \leq \delta, \ \forall i \in \{1, 2, \ldots, |\boldsymbol{\alpha}_{\boldsymbol{w}}(\boldsymbol{x})|\}\}.$$

Since the length of $\boldsymbol{w}$ is $2t$ and its occurrences are non-overlapping, every component in $\boldsymbol{\alpha}_{\boldsymbol{w}}(\boldsymbol{x})$ has value at least $2t$. Furthermore, the number $n_{\boldsymbol{w}}$ of patterns in $\boldsymbol{x}$ is at most $n_{\boldsymbol{w}} \leq \frac{|\boldsymbol{x}|}{2t}$.

For $\boldsymbol{u} \in \Sigma_q^n$, define the VT checksum as $\mathrm{VT}(\boldsymbol{u}) = \sum_{i=1}^n iu_i$. The following construction proposed in [14] applies the VT constraint to the $(\boldsymbol{w}, \delta)$-dense strings, enabling the localization of the deletion to an $\mathcal{O}(\log n)$ interval.

**Lemma 1.** *( [14, Construction 1]) For any integers $0 \leq c_0 < 4$ and $0 \leq c_1 < 2n$, let*

$$\mathcal{C}_{loc}(n, c_0, c_1) \triangleq \{\boldsymbol{x} \in \mathcal{D}_{\boldsymbol{w},\delta}(n) : n_{\boldsymbol{w}}(\boldsymbol{x}) \equiv c_0 \pmod 4, \ \mathrm{VT}(\boldsymbol{\alpha}_{\boldsymbol{w}}(\boldsymbol{x})) \equiv c_1 \pmod{2n}\}.$$

*The code $\mathcal{C}_{loc}(n, c_0, c_1)$ has redundancy $\log n + 4$ and is capable of locating a burst of deletions to an interval of length at most $\delta = t2^{2t+2}\lceil \log n \rceil$.*

In Section III, we give a construction for the first code, which requires roughly $\log n$ bits of redundancy, that is based upon using a simple mapping between non-binary and binary symbols. The more difficult task is to design the second code, which corrects the burst of deletions provided we roughly know its location, given that we want the second code to have redundancy less than $\log n$ bits.

As an illustration of this difficulty, suppose we are provided with the sequence $\boldsymbol{\pi}'$, which is the result of $t$ symbols being deleted from the permutation $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_n)$, and suppose that we know *roughly* where the burst of deletions has occurred, namely in an interval of length $\mathcal{O}(\log n)$. Recall that the constraint in [18] only works for *exactly* $t$ deletions and a set of $t$ related constraints are needed to handle *at most* $t$ deletions. One naive approach is to directly spilt the permutation $\boldsymbol{\pi}$ into several blocks each with length $\mathcal{O}(\log n)$, and introduce constraints into each block. However, the alphabet size for each block is still $n$, which would result in a construction for the second code that cannot have less than $\log n$ bits of redundancy.

In order to avoid this issue, we introduce what we refer to as an "overlapping ranking sequence" for the permutation $\boldsymbol{\pi}$ in Section IV, which allows us to avoid using codes defined over large alphabets. To make use of the connection between the overlapping ranking sequence and its associated permutation, we design codes that are capable of correcting substring edits of length at most $2t$ in Section V. The resulting code, which requires $\mathcal{O}(\log \log n)$ bits of redundancy, is then shown to be capable of correcting a burst of deletions provided that we know its approximate location. The overall construction for correcting a burst of at most $t$ deletions for the permutation code and its total redundancy are presented in Section VI.

## III. Locating the deletion

In this section, we want to identify the location of the burst of deletions to be within an interval of size at most $\mathcal{O}(\log n)$. Our approach will be to first convert each of the permutations in our code to be binary sequences of length $n$

by way of a simple mapping. Afterwards, we will introduce some additional constraints on the resulting binary sequences that will allow us to obtain the desired localizing code.

### A. Mapping from permutations to binary sequences

Define $\boldsymbol{b}_P$: $\mathcal{S}_n \to \Sigma_2^n$ as:

$$b_P(\boldsymbol{\pi})_i = \begin{cases} 1, & \text{if } \pi_i > n/2 \\ 0, & \text{if } \pi_i \leq n/2 \end{cases} \tag{2}$$

**Example 4.** Suppose $\boldsymbol{\pi} = (5, 3, 4, 1, 2, 6)$. Then, the corresponding binary sequence is $\boldsymbol{b}_P(\boldsymbol{\pi}) = (1, 0, 1, 0, 0, 1)$. △

The binary sequence $\boldsymbol{b}_P(\boldsymbol{\pi})$ after mapping will have an equal number of 0s and 1s when $n$ is even, and the number of 1s is one more than 0s when $n$ is odd. For even $n$, let $\mathcal{D}_e(n)$ be the set of binary sequences of length $n$ with equal number of 0s and 1s, and when $n$ is odd, let $\mathcal{D}_e(n)$ be the set of binary sequences that have one more 1s than 0s. We call sequences in $\mathcal{D}_e(n)$ *balanced sequences*. The size of the set $|\mathcal{D}_e(n)|$ is

$$|\mathcal{D}_e(n)| = \begin{cases} \binom{n}{n/2}, & \text{when } n \text{ is even,} \\ \binom{n}{(n+1)/2}, & \text{when } n \text{ is odd.} \end{cases} \tag{3}$$

For simplicity, we will focus on the case where $n$ is even, but similar results also hold for the case where $n$ is odd.

### B. Densifying binary sequences by a fixing pattern $\boldsymbol{w}$

Next, we make use of $(\boldsymbol{w}, \delta)$-dense strings from the set $\mathcal{D}_{\boldsymbol{w},\delta}(n)$ [14], which was introduced in Section II.

The next lemma provides the bound on the size of set $|\mathcal{D}_e(n)|$. The proof is given in Appendix A.

**Lemma 2.** *From Stirling approximation, we have*

$$\frac{2^n\sqrt{6}}{\sqrt{\pi(3n+2)}} \leq |\mathcal{D}_e(n)| = \binom{n}{n/2} \leq \frac{2^{n+1}}{\sqrt{\pi(2n+1)}}.$$

**Lemma 3.** *The number of $(\boldsymbol{w}, \delta)$-dense strings of length $n$ among balanced sequences is*

$$\begin{aligned} |\mathcal{D}_e(n) \cap \mathcal{D}_{\boldsymbol{w},\delta}(n)| &\geq \binom{n}{n/2} - \frac{2^n}{n^{2\log e - 1}} \\ &\geq \frac{2^n\sqrt{6}}{\sqrt{\pi(3n+2)}} - \frac{2^n}{n^{2\log e - 1}}. \end{aligned}$$

*Proof.* Similar to the Proof of Lemma 1 in [14], let $\boldsymbol{z} \in \Sigma_2^n$ and $E_i$ be the event that $\boldsymbol{z}_{[i+1,i+\delta]}$ does not contain the pattern $\boldsymbol{w}$. The probability of $E_i$ is

$$\Pr(E_i) \leq (1 - \frac{1}{2^{2t}})^{\frac{\delta}{2t}} \overset{(a)}{\leq} \frac{1}{n^{2\log e}} \tag{4}$$

where (a) follows from the fact that the function $(1 - 1/x)^x$ is increasing in $x$ for $x > 1$ and $\lim_{x\to\infty}(1 - 1/x)^x = 1/e$. To bound the probability of the event that $\boldsymbol{z} \in \Sigma_2^n$ is not in $\mathcal{D}_{\boldsymbol{w},\delta}$, the union bound yields

$$\Pr(\boldsymbol{z} \notin \mathcal{D}_{\boldsymbol{w},\delta}) \leq (n - \delta + 1)\Pr(E_i) \leq \frac{1}{n^{2\log e - 1}}. \tag{5}$$

Thus,

$$\begin{aligned} |\mathcal{D}_e(n) \cap \mathcal{D}_{\boldsymbol{w},\delta}(n)| &\geq |\mathcal{D}_e(n)| - 2^n \cdot \Pr(\boldsymbol{z} \notin \mathcal{D}_{\boldsymbol{w},\delta}) \\ &\geq \frac{2^n\sqrt{6}}{\sqrt{\pi(3n+2)}} - \frac{2^n}{n^{2\log e - 1}}. \end{aligned} \tag{6}$$

□

Since $\frac{1}{n^{2\log e-1}} = o(\frac{\sqrt{6}}{\sqrt{\pi(3n+2)}})$, the value of $|\mathcal{D}_e(n) \cap \mathcal{D}_{\boldsymbol{w},\delta}(n)|$ is dominated by the first term.

### C. Approximately locating the deletions

**Construction A.** For any integers $0 \leq c_0 < 4$ and $0 \leq c_1 < 2n$, let

$$\begin{aligned} \mathcal{C}_{loc}^P(n, c_0, c_1) \triangleq \{&\boldsymbol{x} \in \Sigma_2^n : \boldsymbol{x} \in \{\mathcal{D}_e(n) \cap \mathcal{D}_{\boldsymbol{w},\delta}(n)\}, \\ &n_{\boldsymbol{w}}(\boldsymbol{x}) \equiv c_0 \bmod 4, \quad \mathrm{VT}(\boldsymbol{\alpha}_{\boldsymbol{w}}(\boldsymbol{x}) \equiv c_1 \bmod 2n\} \end{aligned}$$

**Lemma 4.** *The code $\mathcal{C}_{loc}^P(n, c_0, c_1)$ is capable of locating the burst of deletions to an interval of length at most $\delta = t2^{2t+2}\lceil \log n \rceil$.*

*Proof.* The lemma can be proved from Lemma 1. The only difference is that the binary sequence $\boldsymbol{b}_P(\boldsymbol{\pi})$ is in $\{\mathcal{D}_e(n) \cap \mathcal{D}_{\boldsymbol{w},\delta}(n)\}$. □

It can be noticed that if a burst of at most $t$ deletions occurred in the permutation $\boldsymbol{\pi}$, the corresponding binary sequence $\boldsymbol{b}_P(\boldsymbol{\pi})$ suffers a burst of at most $t$ deletions at the same location. Thus, the localizing code $\mathcal{C}_{loc}^P(n, c_0, c_1)$ also implies the deletion location in $\boldsymbol{\pi}$.

**Lemma 5.** *There exist integers $c_0$ and $c_1$ such that the size of the permutation code whose codewords $\boldsymbol{\pi}$ satisfy $\boldsymbol{b}_P(\boldsymbol{\pi}) \in \mathcal{C}_{loc}^P(n, c_0, c_1)$ is at least $n!/(16n)$.*

*Proof.* By Lemma 2, Lemma 3, and the pigeonhole principle, there exist values of $c_0$ and $c_1$ such that the size of the resulting permutation code with its corresponding binary mapping sequence $\boldsymbol{b}_P(\boldsymbol{\pi}) \in \mathcal{C}_{loc}^P(n, c_0, c_1)$ is at least:

$$\begin{aligned} \frac{|\mathcal{D}_e(n) \cap \mathcal{D}_{\boldsymbol{w},\delta}(n)| \cdot ((n/2)!)^2}{4 \cdot 2n} &\geq \frac{\left(\binom{n}{n/2} - \frac{2^n}{n^{2\log e-1}}\right) \cdot \frac{n!}{\binom{n}{n/2}}}{8n} \\ &\geq \frac{n!(1 - \frac{\sqrt{6\pi}}{3} \cdot n^{1.5-2\log e})}{8n} \\ &\geq \frac{n!}{16n} \end{aligned}$$

where the first inequality follows from Lemma 3 and the second inequality follows from the lower bound in Lemma 2 for $n \geq 2$. The final inequality can also be shown to hold for $n \geq 3$. □

## IV. Mapping the permutation code to the overlapping ranking sequence

In the following, we define a mapping which bears a resemblance to one originally introduced in [18] for the purpose of correcting a burst of deletions when the length of the burst was known. The key difference between their mapping and the one introduced here is that the ranking sequence in [18] was

constructed using disjoint sets of symbols from the underlying permutation whereas our ranking sequence will be generated using every consecutive set of symbols from the underlying permutation. We now describe these ideas in more details.

Note that if $\boldsymbol{\pi}'$ is obtained from $\boldsymbol{\pi}$ through deletions, the identities of the deleted symbols can be determined by noting which symbols are missing.

**Lemma 6.** *Let $\boldsymbol{\pi}' = (\pi'_1, \pi'_2, \ldots, \pi'_{n-t'})$ be obtained from $\boldsymbol{\pi} \in \mathcal{S}_n$ by deleting $t' \leq t$ consecutive symbols. Further, let $\boldsymbol{\pi}'' \in \mathcal{S}_n$ be the result of inserting the deleted symbols consecutively into $\boldsymbol{\pi}'$. For any $\boldsymbol{\pi}'' \neq \boldsymbol{\pi}$, the overlapping ranking sequence $\boldsymbol{p}_{t+1}(\boldsymbol{\pi}'')$ and $\boldsymbol{p}_{t+1}(\boldsymbol{\pi})$ are not identical.*

*Proof.* The lemma is proved by showing that a contradiction arises if we assume that there exist $\boldsymbol{\pi}''$ and $\boldsymbol{\pi}$ such that their corresponding overlapping ranking sequence $\boldsymbol{p}_{t+1}(\boldsymbol{\pi}'')$ and $\boldsymbol{p}_{t+1}(\boldsymbol{\pi})$ are the same.

Suppose the deleted symbols from $\boldsymbol{\pi}$ are $\boldsymbol{\pi}_{[i,i+t'-1]}$ and $\boldsymbol{\pi}''$ is the result of inserting these symbols (consecutively) beginning at position $j$. Without loss generality, we assume that $j < i$.

Thus, $\boldsymbol{\pi}$ and $\boldsymbol{\pi}''$ can be shown as:

$$\boldsymbol{\pi} = (\cdots, \pi_j, \pi_{j+1}, \cdots, \pi_{i-1}, \pi_i, \cdots, \pi_{i+t'-1}, \cdots)$$
$$\boldsymbol{\pi}'' = (\cdots, \pi''_j, \pi''_{j+1}, \cdots, \pi''_{i-1}, \pi''_i, \cdots, \pi''_{i+t'-1}, \cdots)$$

From the definition of $\boldsymbol{\pi}''$, we can have $\pi''_k = \pi_{k-t'}$ when $k \geq j + t'$. To illustrate the changed and unchanged part in $\boldsymbol{\pi}$ and $\boldsymbol{\pi}''$, we denote the unchanged part in both $\boldsymbol{\pi}$ and $\boldsymbol{\pi}''$ as $(\pi_j, \pi_{j+1}, \ldots, \pi_{i-1}) = (\pi''_{j+t'}, \pi''_{j+t'+1}, \ldots, \pi''_{i+t'-1}) = (a_1, a_2, \ldots, a_m)$, where $m = i - j$. Further, we use $(x_1, x_2, \ldots, x_{t'}) = (\pi_i, \pi_{i+1}, \ldots, \pi_{i+t'-1})$ to denote the deleted symbols from $\boldsymbol{\pi}$ and $(x''_1, x''_2, \ldots, x''_{t'})$ to denote the inserted symbols in $\boldsymbol{\pi}''$, where $(x''_1, x''_2, \ldots, x''_{t'})$ is a permutation of the deleted symbols $(x_1, x_2, \ldots, x_{t'})$. Then, $\boldsymbol{\pi}_{[j,i+t'-1]}$ and $\boldsymbol{\pi}''_{[j,i+t'-1]}$ can be rewritten as the following, taking $m > t'$ case as example:

$$\boldsymbol{\pi} = (a_1, a_2, \cdots, a_{t'}, a_{t'+1}, \cdots, a_m, x_1, \cdots, x_{t'})$$
$$\boldsymbol{\pi}'' = (x''_1, x''_2, \cdots, x''_{t'}, a_1, \cdots, a_{m-t'}, a_{m-t'+1}, \cdots, a_m)$$

For uniformity of notation, we sometimes denote $x_j$ by $a_{m+j}$ and $x''_j$ by $a_{-t'+j}$. Let $[y_a, y_b] = \{y_a, y_{a+1}, \ldots, y_b\}$. For a set $\{y_1, y_2, \ldots, y_k\}$ with distinct elements, we say $a \prec \{y_1, y_2, \ldots, y_k\}$ if $a \leq y_i$ holds for all $i \in \{1, \ldots, k\}$, with equality holding for at most one value of $i$.

- Consider the case where $i - j > t$. To guarantee each element in $\boldsymbol{p}_{t+1}(\boldsymbol{\pi})$ and $\boldsymbol{p}_{t+1}(\boldsymbol{\pi}'')$ are the same, we can have the following two relationships:
  If $a_i \prec [a_i, a_{i+t'}]$, for $1 \leq i \leq m - t'$, then
  $$a_{i+t'} \prec [a_{i+t'}, a_{i+2t'}] \tag{7}$$
  and if $a_i \prec [a_i, a_m]$, for $m - t' < i < m$, then
  $$a_{i+t'} \prec [a_{i+t'}, a_{m+t'}] \tag{8}$$
  Let $a^* = \min\{a_1, \ldots, a_m, x_1, \ldots, x_{t'}\} = \min\{x''_1, \ldots, x''_{t'}, a_1, \ldots, a_m\}$. Recall that all elements are distinct.
  Suppose there exists some $1 \leq i \leq m$ such that $a^* = a_i$. Note that elements $[a_i, a_{i+t'}]$ in $\boldsymbol{\pi}$ and $[a_{i-t'}, a_i]$ in $\boldsymbol{\pi}''$ have the same value in the corresponding overlapping ranking sequence as the number of elements in each of these segments is $t' + 1 \leq t + 1$. On the other hand, if the minimum $a_i$ appears in two different places in a segment with same starting and end location in $\boldsymbol{\pi}$ and $\boldsymbol{\pi}''$, then the overlapping ranking sequence cannot be the same. Thus, it implies the contradiction arises when considering the minimum element $a^*$ in $\{a_1, \ldots, a_m\}$.
  Hence, there must be some $1 \leq i \leq t'$ such that $a^* = x''_i$. Noting that $r(a_i, \ldots, a_{i+t'}) = r(x''_i, \ldots, x''_{t'}, a_1, \ldots, a_i)$ and we have $a_i \prec [a_i, a_{i+t'}]$. We now show that
  $$\begin{aligned} a_i &\prec [a_i, a_{i+t'}] \\ a_{i+t'} &\prec [a_{i+t'}, a_{i+2t'}] \\ a_{i+2t'} &\prec [a_{i+2t'}, a_{i+3t'}] \\ &\vdots \\ a_{i+kt'} &\prec [a_{i+kt'}, a_{i+kt'+t'}] \\ a_{i+kt'+t'} &\prec [a_{i+kt'+t'}, a_{m+t'}] \end{aligned} \tag{9}$$
  where $k$ is the largest integer such that $i + kt' \leq m$. All relations except the last one follow from (7) and the last one follows from (8). The last two relations imply that
  $$a_{i+kt'} \prec [a_{i+kt'}, a_{m+t'}] = (a_{i+kt'}, \ldots, a_m, x_1, \ldots, x_{t'}),$$
  which is a contradiction since the minimum among all elements is among the elements $\{x''_1, \ldots, x''_{t'}\}$, where $\{x''_1, \ldots, x''_{t'}\}$ and $\{x_1, \ldots, x_{t'}\}$ contain the same elements.
- Consider the case where $i - j \leq t$. When $j \neq i$, note that the elements $[a_m, a_{m+t'}]$ in $\boldsymbol{\pi}$ and $[a_{m-t'}, a_m]$ in $\boldsymbol{\pi}''$ have the same value in the corresponding overlapping ranking sequence as the number of elements in each of these segments is $t' + 1 \leq t + 1$. Also, $[a_m, a_{m+t'}]$ in $\boldsymbol{\pi}$ and $[a_{m-t'}, a_m]$ in $\boldsymbol{\pi}''$ have common elements $a_m$ and $x_i, \exists i \in \{1, \ldots, t'\}$. However, this is impossible that $a_m$ and $x_i$ are in reversed order in both.
  When $i = j$, the elements in $\boldsymbol{\pi}_{[i,i+t]}$ and $\boldsymbol{\pi}''_{[i,i+t]}$ cannot be in the same order due to $\boldsymbol{\pi}'' \neq \boldsymbol{\pi}$. Thus, the overlapping ranking sequence $\boldsymbol{p}_{t+1}$ of $\boldsymbol{\pi}$ and $\boldsymbol{\pi}''$ are not the same in this case. □

After mapping the permutation code to the overlapping ranking sequence, the alphabet size is reduced from $n$ to $(t+1)!$. As a result, we want to correct a burst of deletions in $\boldsymbol{\pi}$ by first recovering the corresponding overlapping ranking sequence $\boldsymbol{p}_{t+1}(\boldsymbol{\pi})$, and then we will use this information to uniquely determine $\boldsymbol{\pi}$ according to Lemma 6. Recall that for a string $(v_1, v_2, \ldots, v_n)$, we say that $(v_i, v_{i+1}, \ldots, v_{i+\ell-1})$ is a substring of length $\ell$ that appears in $(v_1, v_2, \ldots, v_n)$ at position $i$.

**Claim 1.** *After deleting a burst of at most $t$ symbols in a permutation $\boldsymbol{\pi}$ resulting in $\boldsymbol{\pi}'$, the corresponding overlapping ranking sequence $\boldsymbol{p}_{t+1}(\boldsymbol{\pi}')$ can be obtained from $\boldsymbol{p}_{t+1}(\boldsymbol{\pi})$ by*

*at most $t$ consecutive substitutions followed by a burst of at most $t$ deletions.*

*Proof.* We can write $\boldsymbol{p}_{t+1}(\boldsymbol{\pi})$ as:

$$\boldsymbol{p}_{t+1}(\boldsymbol{\pi}) = (p_1, p_2, \ldots, p_{i-t}, \ldots, p_{i-1}, p_i, p_{i+1}, \ldots, p_{i+t'-1}, p_{i+t'}, \ldots) \tag{10}$$

Let $\boldsymbol{\pi}'$ be the result of deleting $t' \leq t$ consecutive symbols $\boldsymbol{\pi}_{[i,i+t'-1]}$ from the permutation $\boldsymbol{\pi}$. Thus, the corresponding $\boldsymbol{p}_{t+1}(\boldsymbol{\pi}')$ of permutation $\boldsymbol{\pi}'$ can be written as:

$$\boldsymbol{p}_{t+1}(\boldsymbol{\pi}') = \big(p_1, p_2, \ldots, p'_{i-t}, \ldots, p'_{i-1}, p_{i+t'}, \ldots\big) \tag{11}$$

where we use $p_i$ to denote an unchanged value and $p'_j$ to denote a possibly changed value in $\boldsymbol{p}_{t+1}(\boldsymbol{\pi}')$ compared with $\boldsymbol{p}_{t+1}(\boldsymbol{\pi})$.

By comparing (11) with (10), we see that there are at most $t$ consecutive substitutions (substituting $(p_{i-t}, \ldots, p_{i-1})$ by $(p'_{i-t}, \ldots, p'_{i-1})$) followed by at most $t$ consecutive deletions (deleting $(p_i, \ldots, p_{i+t'-1})$). □

Based on this observation, we characterize this error pattern as substring edits that replace a substring of length at most $2t$ with another substring of length at most $2t$, which is a more general type of error. Thus, in the next subsection, we will discuss how to construct codes capable of correcting substring edits of length at most $2t$ for recovering the overlapping ranking sequence $\boldsymbol{p}_{t+1}(\boldsymbol{\pi})$.

## V. Correcting substring edits of length at most $2t$ in the overlapping ranking sequence

In this section, our goal is to construct a code for correcting substring edits of length at most $2t$ in the overlapping ranking sequence $\boldsymbol{p}_{t+1}(\boldsymbol{\pi})$ based on the systematic binary code capable of correcting up to $t$ edits [19].

For $q < \log n$, the basic idea is to consider $q$-ary sequences as a set of $\lceil \log q \rceil$ binary sequences. Then, we choose the systematic binary code proposed in [19] as the base code, which is capable of correcting up to $t$ edits, where each edit is a deletion, insertion or substitution error. It is straightforward to see that the number of edits for substring edits of length at most $2t$ is also at most $2t$. Thus, we should set the number of edits to $2t$ in our problem.

**Lemma 7.** *(c.f., [19]) Let $t$ be a constant with respect to $k$. There exist an integer $a \leq 2^{4t\log k + o(\log k)}$ and a labeling function $f_{2t} : \Sigma_2^k \to \Sigma_{2^{\mathcal{R}_{2t}(k)}}$, where $\mathcal{R}_{2t}(k) = O(t^4 \log k)$ such that $\{(\boldsymbol{x}, a, f_{2t}(\boldsymbol{x}) \bmod a) : \boldsymbol{x} \in \Sigma_2^k\}$ can correct substring edits of length at most $2t$.*

Therefore, we can get the following lemma for $q$-ary sequences:

**Lemma 8.** *Let $t$ be a constant with respect to $k$. There exist an integer $a_q \leq 2^{\lceil \log q \rceil (4t \log k + o(\log k))}$ and a labeling function $f_{2t}^q : \Sigma_q^k \to \Sigma_{2^{\lceil \log q \rceil \mathcal{R}_{2t}(k)}}$, where $f_{2t}^q(\boldsymbol{u}) = \sum_{i=1}^{\lceil \log q \rceil} 2^{\mathcal{R}_{2t}(k)(i-1)} f_{2t}(A(\boldsymbol{u})_i)$ such that $\{(\boldsymbol{u}, a_q, f_{2t}^q(\boldsymbol{u}) \bmod a_q) : \boldsymbol{u} \in \Sigma_q^k\}$ can correct substring edits of length at most $2t$ in $q$-ary sequences.*

From Lemma 4, we can narrow the deletion to an $\mathcal{O}(\log n)$ interval in the permutation $\boldsymbol{\pi}$. Then, we will make use of Lemma 8 to construct a code for correcting substring edits of length at most $2t$ in the corresponding overlapping ranking sequence $\boldsymbol{p}_{t+1}(\boldsymbol{\pi})$ with this positional knowledge (We omit the argument $t+1$ and $\boldsymbol{\pi}$ from $\boldsymbol{p}_{t+1}(\boldsymbol{\pi})$ and simply write $\boldsymbol{p}$ in the rest of this subsection).

We split the sequence $\boldsymbol{p}$ into two sets $\boldsymbol{p}_e = \{\boldsymbol{p}_{e,1}, \boldsymbol{p}_{e,2}, \ldots, \boldsymbol{p}_{e,s}\}$ and $\boldsymbol{p}_o = \{\boldsymbol{p}_{o,1}, \boldsymbol{p}_{o,2}, \ldots, \boldsymbol{p}_{o,s+1}\}$, where $s = n/2P$ and $P = t2^{2t+2}\lceil \log n \rceil$ for even and odd blocks, respectively:

- **Even Blocks**: $\boldsymbol{p}_{e,i} = \boldsymbol{p}_{[(2i-2)P+1, 2iP]},\ i = 1, \cdots, s$
- **Odd Blocks**:

$$\boldsymbol{p}_{o,i} = \begin{cases} \boldsymbol{p}_{[1,P]}, & i = 1; \\ \boldsymbol{p}_{[(2i-3)P+1,(2i-1)P]}, & i = 2, \cdots, s; \\ \boldsymbol{p}_{[n-P+1,n]}, & i = s+1. \end{cases}$$

For $i \in [s]$, let $a_{e,i}^q = E_{tB}(\boldsymbol{p}_{e,i})$ and similarly let $a_{o,i}^q = E_{tB}(\boldsymbol{p}_{o,i})$ for $i \in [s+1]$. Note that $\boldsymbol{p}_e$ and $\boldsymbol{p}_o$ each cover the sequence $\boldsymbol{p}$ and that any interval of length $P$ is fully contained in at least one block in $\boldsymbol{p}_e$ or in $\boldsymbol{p}_o$. We can use the $E_{SB}$ to protect each block of length $2P$, as in the following lemma.

**Lemma 9.** *There exists an integer $a = 2^{\lceil \log q \rceil (4t \log(2P) + o(\log P))}$ such that for any $d_1, e_1 \in [[a]]$, $d_2, e_2 \in [[a]]$, the code $\mathcal{C}_{2t}(n, t, P)$ such that*

$$\mathcal{C}_{2t}(n,t,P) = \{\boldsymbol{p} \in \Sigma_{(t+1)!}^n : \sum_{i=1}^{s} a_{e,i}^q = d_1 \bmod a, \sum_{i=1}^{s} \big(f_{2t}^q(\boldsymbol{p}_{e,i}) \bmod a_{e,i}^q\big) = e_1 \bmod a, \sum_{i=1}^{s+1} a_{o,i}^q = d_2 \bmod a, \sum_{i=1}^{s+1} \big(f_{2t}^q(\boldsymbol{p}_{o,i}) \bmod a_{o,i}^q\big) = e_2 \bmod a\}.$$

*can correct one substring edit of length at most $2t$ with the knowledge that the location of the edited symbols is within $P$ consecutive positions. Furthermore, there exist choices for $d_1, d_2$ and $e_1, e_2$ such that the redundancy of the code is at most $4 \log a$.*

*Proof.* The interval of length $P$ in which the edit has occurred is fully contained in a block of $\boldsymbol{p}_e$ or in a block of $\boldsymbol{p}_o$. Without loss of generality, let us assume the former and also assume that the index of this block is $l$. We can recover all blocks of $\boldsymbol{p}_e$ except $\boldsymbol{p}_{e,l}$. The value of $a_{e,l}^q$ and $f_{2t}^q(\boldsymbol{p}_{e,l}) \bmod a_{e,l}^q$ can be determined by solving the equation $\sum_{i=1}^{s} a_{e,i}^q \equiv d_1 \bmod a$ and $\sum_{i=1}^{s} (f_{2t}^q(\boldsymbol{p}_{e,i}) \bmod a_{e,i}^q) \equiv e_1 \bmod a$, respectively. Then, by Lemma 9, the block $\boldsymbol{p}_{e,l}$ can be recovered. □

## VI. Overall construction

Building on the previous sections, we can present the overall construction of the permutation code for correcting a burst of at most $t$ deletions. First, we apply the code $\mathcal{C}_{loc}^P(n, c_0, c_1)$ to narrow the deletion into an interval of length $t2^{2t+2}\lceil \log n \rceil$ with redundancy $\log n + \mathcal{O}(1)$. Then, we recover the permutation via $\mathcal{C}_{2t}(n, (t+1)!, t2^{2t+2}\lceil \log n \rceil)$ for correcting the corresponding overlapping ranking sequence with the positional knowledge of the deletion.

**Construction B.** There exists an integer $a = 2^{\lceil \log(t+1)! \rceil (4t \log\log n + o(\log\log n))}$ such that for all $c_0 \in [[4]]$, $c_1 \in [[2n]]$, $d_1, d_2 \in [[a]]$ and $e_1, e_2 \in [[a]]$., we define a permutation code $\mathcal{P}_{\leq t}(n)$ over $\mathcal{S}_n$ as

$$\mathcal{P}_{\leq t}(n) \triangleq \{\boldsymbol{\pi} \in \mathcal{S}_n : \boldsymbol{b}_P(\boldsymbol{\pi}) \in \mathcal{C}_{loc}^P(n, c_0, c_1), \\ \boldsymbol{p}_{t+1}(\boldsymbol{\pi}) \in \mathcal{C}_{2t}(n, (t+1)!, t2^{2t+2}\lceil \log n \rceil)\}$$

**Theorem 1.** *The permutation code $\mathcal{P}_{\leq t}(n)$ over $\mathcal{S}_n$ is capable of correcting a burst of at most $t$ deletions with the redundancy at most $\log n + \mathcal{O}(\log\log n)$ bits.*

*Proof.* The error-correcting capability of the code has already been discussed. From Lemma 9, the redundancy of the second part in the permutation code $\mathcal{P}_{\leq t}(n)$ for correcting overlapping ranking sequence will be $4 \log a$. Since $P = t2^{2t+2}\lceil \log n \rceil$ and $t$ is a constant, we have

$$4 \log a = \mathcal{O}(\log\log n).$$

Combining with Lemma 5, the code size $|\mathcal{P}_{\leq t}(n)|$ is at least

$$|\mathcal{P}_{\leq t}(n)| = \frac{n!}{16n \cdot a^4} \geq \frac{n!}{16n \cdot 2^{\mathcal{O}(\log\log n)}}.$$

Therefore, the total redundancy of the permutation code $\mathcal{P}_{\leq t}(n)$ is at most $\log n + \mathcal{O}(\log\log n)$. □

## VII. Conclusion

In this paper, we presented permutation codes capable of correcting a burst of at most $t$ deletions, which greatly improves upon the state-of-the-art in terms of the redundancy. However, there still remain some interesting problems, including extending this work to multipermutations and multiple bursts of deletions.

## References


[1] D. Slepian, "Permutation modulation," *Proceedings of the IEEE*, vol. 53, no. 3, pp. 228–236, 1965.

[2] W. Chu, C. J. Colbourn, and P. Dukes, "Constructions for permutation codes in powerline communications," *Designs, Codes and Cryptography*, vol. 32, no. 1, pp. 51–64, 2004.

[3] A. MACK, "Coded modulation for powerline communications," *AEU Int. J. Eletron. Commun.*, vol. 54, no. 1, pp. 45–49, 2000.

[4] D. De la Torre, C. Colbourn, and A. Ling, "An application of permutation arrays to block ciphers," *Congressus Numerantium*, pp. 5–8, 2000.

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

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

[7] H. Zhou, M. Schwartz, A. A. Jiang, and J. Bruck, "Systematic error-correcting codes for rank modulation," *IEEE Transactions on Information Theory*, vol. 61, no. 1, pp. 17–32, 2014.

[8] F. Farnoud, V. Skachek, and O. Milenkovic, "Error-correction in flash memories via codes in the ulam metric," *IEEE Transactions on Information Theory*, vol. 59, no. 5, pp. 3003–3020, 2013.

[9] A. Barg and A. Mazumdar, "Codes in permutations and error correction for rank modulation," *IEEE Transactions on Information Theory*, vol. 56, no. 7, pp. 3158–3165, 2010.

[10] R. Gabrys, E. Yaakobi, F. Farnoud, F. Sala, J. Bruck, and L. Dolecek, "Codes correcting erasures and deletions for rank modulation," *IEEE Transactions on Information Theory*, vol. 62, no. 1, pp. 136–150, 2015.

[11] F. Sala, R. Gabrys, and L. Dolecek, "Deletions in multipermutations," in *2014 IEEE International Symposium on Information Theory*. IEEE, 2014, pp. 2769–2773.

[12] C. Schoeny, A. Wachter-Zeh, R. Gabrys, and E. Yaakobi, "Codes correcting a burst of deletions or insertions," *IEEE Transactions on Information Theory*, vol. 63, no. 4, pp. 1971–1985, 2017.

[13] J. Sima, R. Gabrys, and J. Bruck, "Syndrome compression for optimal redundancy codes," in *2020 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2020, pp. 751–756.

[14] A. Lenz and N. Polyanskii, "Optimal codes correcting a burst of deletions of variable length," in *International Symposium on Information Theory (ISIT)*, 2020.

[15] H. Han, J. Mu, X. Jiao, and Y.-C. He, "Codes correcting a burst of deletions for permutations and multi-permutations," *IEEE Communications Letters*, vol. 22, no. 10, pp. 1968–1971, 2018.

[16] H. Han, J. Mu, Y.-C. He, X. Jiao, and W. Ma, "Multi-permutation codes correcting a single burst unstable deletions in flash memory," *IEEE Communications Letters*, vol. 24, no. 4, pp. 720–724, 2020.

[17] ——, "Rate-improved permutation codes for correcting a single burst of deletions," *IEEE Communications Letters*, vol. 25, no. 1, pp. 49–53, 2020.

[18] Y. M. Chee, S. Ling, T. T. Nguyen, H. Wei, X. Zhang *et al.*, "Burst-deletion-correcting codes for permutations and multipermutations," *IEEE Transactions on Information Theory*, vol. 66, no. 2, pp. 957–969, 2019.

[19] J. Sima, R. Gabrys, and J. Bruck, "Optimal systematic t-deletion correcting codes," in *2020 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2020, pp. 769–774.


## Appendix A
## Proof of Lemma 2

**Lemma 2.** *From Stirling approximation, we have*

$$\frac{2^n\sqrt{6}}{\sqrt{\pi(3n+2)}} \leq |\mathcal{D}_e(n)| = \binom{n}{n/2} \leq \frac{2^{n+1}}{\sqrt{\pi(2n+1)}}.$$

*Proof.* First, we have

$$\left(\frac{n+\frac{1}{2}}{n+1}\right)^2 = \frac{n^2+n+\frac{1}{4}}{n^2+2n+1} \leq \frac{n+\frac{1}{3}}{n+\frac{4}{3}}.$$

Thus,

$$\frac{\binom{2n+2}{n+1}}{\binom{2n}{n}} = 4\frac{n+\frac{1}{2}}{n+1} \leq 4\sqrt{\frac{n+\frac{1}{3}}{n+\frac{4}{3}}}.$$

which implies $\binom{2n}{n}\frac{n+1/3}{4^n}$ is decreasing.

We also have

$$\left(\frac{n+\frac{1}{2}}{n+1}\right)^2 = \frac{n^2+n+\frac{1}{4}}{n^2+2n+1} \leq \frac{n+\frac{1}{4}}{n+\frac{5}{4}}.$$

Then,

$$\frac{\binom{2n+2}{n+1}}{\binom{2n}{n}} = 4\frac{n+\frac{1}{2}}{n+1} \leq 4\sqrt{\frac{n+\frac{1}{4}}{n+\frac{5}{4}}}.$$

which implies $\binom{2n}{n}\frac{n+1/4}{4^n}$ is increasing.

From

$$\lim_{n\to\infty} \frac{\sqrt{\pi n}}{4^n}\binom{2n}{n} = 1$$

We can show that

$$\frac{4^n}{\sqrt{\pi\left(n+\frac{1}{3}\right)}} \leq \binom{2n}{n} \leq \frac{4^n}{\sqrt{\pi\left(n+\frac{1}{4}\right)}}.$$

which implies

$$\frac{2^n\sqrt{6}}{\sqrt{\pi(3n+2)}} \leq \binom{n}{n/2} \leq \frac{2^{n+1}}{\sqrt{\pi(2n+1)}}.$$

□