Limited-Magnitude Error-Correcting Gray Codes for Rank Modulation

Yonatan Yehezkeally, *Student Member, IEEE,* Moshe Schwartz, *Senior Member, IEEE*

***Abstract*—We construct error-correcting codes over permutations under the infinity-metric, which are also Gray codes in the context of rank modulation, i.e., are generated as simple circuits in the rotator graph. These errors model limited-magnitude or spike errors, for which only single-error-detecting Gray codes are currently known. Surprisingly, the error-correcting codes we construct achieve a better asymptotic rate than that of presently known constructions not having the Gray property, and exceed the Gilbert-Varshamov bound. Additionally, we present efficient ranking and unranking procedures, as well as a decoding procedure that runs in linear time. Finally, we also apply our methods to solve an outstanding issue with error-detecting rank-modulation Gray codes (also known in this context as snake-in-the-box codes) under a different metric, the Kendall τ-metric, in the group of permutations over an even number of elements S_{2n}, where we provide asymptotically optimal codes.**

***Index Terms*—Gray codes, error-correcting codes, permutations, spread-d circuit codes, rank modulation**

I. Introduction

RANK modulation is a method for storing information in non-volatile memories [23], which has been researched in recent years. It calls for encoding information in relative values in a group of cells rather than the absolute values of each single cell. More precisely, it stores information in the permutation suggested by sorting a group of cells by their relative values; Such values may be charge levels in flash memory cells or electrical resistance in phase-change memory [30]. Rank modulation allows for increased robustness against certain noise mechanisms (e.g., charge leakage in flash memory cells), as well as alleviating some inherent challenges in flash memories (e.g., programming/erasure-asymmetry and programming-overshoot). Permutation codes in general have also previously seen usages in source-encoding [3]–[5], [38] and signal detection [7], as well as other fields [6], [9], [11], and more recently been used in power-line communications [41].

Several error models have been studied for rank modulation, including the Kendall τ-metric [2], [24], [28], [47], the ℓ_∞-metric [27], [36], [39], [40] and other examples [12], [18]. In this paper we focus on the ℓ_∞-metric, which models limited-magnitude or spike noise, i.e., we assume that the rank of any given cell–its position when sorting the group of cells–could not have changed by more than a given amount. [27], [39] have presented constructions for error-correcting codes under this metric, as well as explored some non-constructive lower- and upper-bounds on the parameters of existing codes. [40] has since employed methods of relabeling to optimize the minimal distance of known constructions.

In the context of rank modulation, a generalization of the Gray code has been shown to reduce write-time–by eliminating the risk of programming-overshoot–and allow integration with other multilevel-cells coding schemes [13], [14], [23]. Gray codes were first considered over the space of binary vectors, where they were generally defined as a listing of distinct vectors–sometimes exhaustive–such that each pair of consecutive vectors differed by a single bit-flip [19]; the concept has since been generalized in some contexts to include codes over arbitrary alphabets, requiring only that codewords could be ordered in a sequence, where each codeword is derived from the previous by one of a predefined set of functions. Put differently, Gray codes may be considered as simple paths on the digraph whose nodes are elements of the alphabet, and edges are induced by the aforementioned functions set (e.g., Cayley graphs). Suggested usages of Gray codes in contexts other than rank modulation, surveyed in [33], include permanent-computation [29], circuit-testing [31], image-processing [1], hashing [17], coding [15], [23], [34] and data storing/extraction [8]. Within rank modulation, particular Cayley graphs were used, which were first proposed (for use in multiprocessor networks) in [10], [16] as Faber–Moore- or rotator graphs, and later rediscovered (the authors being apparently unaware) for use in Flash memories in [23] (including one of its constructions). These codes are in fact also an example of greedily constructed Gray codes [43].

Gray codes with error-correction capabilities have sometimes been referred to as spread-d circuit codes (see [21] and references therein). Specifically, in the context of rank modulation, such codes were so far only studied for the case of single-error detection, where they were dubbed (see [20]–[22], [42], [44]–[46]) snake-in-the-box codes (or, more appropriately, coil-in-the-box codes, when they are cyclic); this, again, draws on terminology first used with Gray codes in the hypercube, where snake-in-the-box codes are defined as spread-2 codes using the Hamming distance [37]. [44] studied such rank-modulation codes under both the Kendall τ-metric and the ℓ_∞-metric, and more recent papers [20], [22], [45] have categorized and constructed optimally sized coil-in-the-box codes under the former metric for odd orders, although the case of even orders proved more challenging (see [46] in

This paper was presented in part at the 2016 IEEE International Symposium on Information Theory.

The authors are with the Department of Electrical and Computer Engineering, Ben-Gurion University of the Negev, Beer-Sheva 8410501, Israel (e-mail: yonatany@post.bgu.ac.il; schwartz@ee.bgu.ac.il).

This work was supported in part by ISF grant no. 130/14.

addition to the aforementioned papers).

In this work we focus on the ℓ_∞-metric and present a construction of error-correcting Gray codes capable of correcting an arbitrary number of limited-magnitude errors. The allowed transitions between codewords are the "push-to-the-top" operations, used in most previous works [13], [14], [20], [22], [23], [44], [45] (which are isomorphic to the prefix-rotations of [10]). An example of such a code, generated in the paper in Example 19, is presented in Figure 1. The resulting codes will be shown to have greater size than known constructions in the case of fixed minimal distance, as well as achieve better asymptotic rates than known codes in the case of $d = \Theta(n)$; both size and rate are also compared against known bounds. In particular, in the case of error-detecting codes (i.e., $d = 2$), where some codes with the Gray property are already known (that is, snake-in-the-box codes, developed in [44]), our new construction outperforms known codes by a factor of $\sim \frac{n^2}{8}$ (we note that in this specific case of $d = 2$, an equivalent construction was also concurrently published in a preprint in [42]).

We will also briefly examine error-detecting codes under the Kendall τ-metric for even orders, since methods developed for the application of our main construction can readily be adapted to that purpose. We provide an asymptotically optimal construction which nearly completes the categorization of available codes for that scheme.

The paper is organized as follows. In Section II we present notations and definitions. In Section III we study a new kind of auxiliary codes which are required for our construction, before presenting it in Section IV and discussing its performance in comparison with known constructions and bounds. We devise a decoding algorithm for the generated codes in Section V, and discuss ranking and unranking procedures in Section VI. We briefly present an adaptation of the developed auxiliary codes to error-detecting codes under the Kendall τ-metric in Section VII. Finally, we conclude in Section VIII by reviewing our results and suggesting problems for future study.

II. Preliminaries

For $n \in \mathbb{N}$, we let S_n denote the *symmetric group*, the set of all permutations on $[n] = \{1, 2, \dots, n\}$ (i.e., bijections $\sigma : [n] \xrightarrow[\text{onto}]{1-1} [n]$), with composition as group action:

$$\sigma\tau(k) = (\sigma \circ \tau)(k) = \sigma(\tau(k)).$$

Throughout the paper we shall denote the identity permutation $\mathrm{Id} \in S_n$ defined for all $k \in [n]$: $\mathrm{Id}(k) = k$.

We use the *cycle notation* for permutations, i.e., for distinct $\left\{a_j\right\}_{j=1}^{k} \subseteq [n]$ we let $\sigma = (a_1, a_2, \dots, a_k)$ be the permutation such that $\sigma(a_j) = a_{(j \bmod k)+1}$ and $\sigma(b) = b$ for all $b \in [n] \setminus \left\{a_j\right\}_{j=1}^{k}$. Trivially, every permutation can be represented as a composition of disjoint cycles. It is also well known that every permutation can be represented as a composition of *transpositions*, cycles of length 2, and that the parity of the number of transpositions in that representation is unique (although the representation itself is not). We therefore have *even* and *odd* permutations, and we use $\mathrm{sign}(\sigma) = \pm 1$ to indicate the parity of the number of elements in transposition representation (with no particular importance given to which is which). The set of even permutations forms a subgroup $A_n \leqslant S_n$ named the *alternating group*. We will say that $C \subseteq S_n$ is *parity-preserving* if every two elements $\sigma, \tau \in C$ have the same parity, that is, $\mathrm{sign}\,\sigma = \mathrm{sign}\,\tau$ (put differently, either $C \subseteq A_n$ or $C \subseteq S_n \setminus A_n$).

We also use the *vector notation* for permutations,

$$\sigma = [\sigma(1),\ \sigma(2),\ \dots\ , \sigma(n)] .$$

This allows us to more easily notate, for $1 \leqslant i < j \leqslant n$, the *"push-to-the-ith-index"* transition $t_{i\uparrow j} : S_n \to S_n$ by

$$\begin{aligned} t_{i\uparrow j}&\left(\left[a_1, a_2, \dots, a_{i-1}, a_i, a_{i+1}, \dots, a_{j-1}, a_j, a_{j+1}, \dots, a_n\right]\right) \\ &= \left[a_1, a_2, \dots, a_{i-1}, a_j, a_i, a_{i+1}, \dots, a_{j-1}, a_{j+1}, \dots, a_n\right] . \end{aligned}$$

We follow recent works [22], [23], [44], [45] (among others) in dubbing "push-to-the-1st-index" transitions as *"push-to-the-top"* transitions (although these operations–or more precisely their inverse–were originally introduced as prefix-rotations in [10]); We denote $t_{\uparrow j} = t_{1\uparrow j}$. Finally, we define the *"push-to-the-bottom"* transition on the jth index, $t_{\downarrow j} : S_n \to S_n$,

$$\begin{aligned} t_{\downarrow j}&\left(\left[a_1, a_2, \dots, a_{j-1}, a_j, a_{j+1}, \dots, a_n\right]\right) \\ &= \left[a_1, a_2, \dots, a_{j-1}, a_{j+1}, \dots, a_n, a_j\right] . \end{aligned}$$

Given any set S, and a collection of transitions

$$T \subseteq \{f \mid f : S \to S\} ,$$

we define a *T-Gray code* over the set S to be a sequence $C = (c_r)_{r=1}^{M} \subseteq S$ such that for all $1 \leqslant r < r' \leqslant M$ we have $c_r \neq c_{r'}$ and such that for all $1 \leqslant r < M$ there exists $t_r \in T$ satisfying $c_{r+1} = t_r(c_r)$. We say that a sequence C is contained in S, by abuse of notation, if $c_r \in S$ for all r. That is, we may refer to a Gray code as an unordered set–or simply a code–when desired for simplicity. Conversely, we say that a code has the *Gray property*, or is a Gray code, if it can be so ordered. We call $M = |C|$ the *size* of the code, and $t_1, t_2, \dots, t_{M-1}$ the transition sequence *generating* C. If there exists $t \in T$ such that $c_1 = t(c_M)$ we say that C is *cyclic*, and include $t_M = t$ in its generating transition sequence. If $C = S$, we say that C is a *complete* code.

Example 1 In the classic example of a Gray code we have, e.g., $S = \mathbb{F}_2{}^3$, with T consisting of the group action of $\{001, 010, 100\} \subseteq S$ on S, defined

$$v(u) = u + v.$$

We then have the complete cyclic Gray code given by

$$\begin{tikzcd} 000 \arrow[r, "001"] & 001 \arrow[r, "010"] & 011 \arrow[r, "001"] & 010 \arrow[d, "100"] \\ 100 \arrow[u, "100"] & 101 \arrow[l, "001"'] & 111 \arrow[l, "010"'] & 110 \arrow[l, "001"'] \end{tikzcd}$$

□

In this paper, we fix $S = S_n$. Since we intend to work with minutely distinct classes of codes on the symmetric group in this paper, we will introduce notations to distinguish them, which are organized in Table I for the readers' comfort. We say that $C = (c_1, c_2, \dots, c_M) \subseteq S_n$ is a $G_{i\uparrow}(n, M)$ if it is a cyclic

$$\begin{bmatrix}4\\1\\5\\2\\6\\3\end{bmatrix}\xrightarrow{t_{\uparrow 6}}\begin{bmatrix}3\\4\\1\\5\\2\\6\end{bmatrix}\xrightarrow{t_{\uparrow 3}}\begin{bmatrix}1\\3\\4\\5\\2\\6\end{bmatrix}\xrightarrow{t_{\uparrow 3}}\begin{bmatrix}4\\1\\3\\5\\2\\6\end{bmatrix}\xrightarrow{t_{\uparrow 5}}\begin{bmatrix}2\\4\\1\\3\\5\\6\end{bmatrix}\xrightarrow{t_{\uparrow 3}}\begin{bmatrix}1\\2\\4\\3\\5\\6\end{bmatrix}\xrightarrow{t_{\uparrow 3}}\begin{bmatrix}4\\1\\2\\3\\5\\6\end{bmatrix}\xrightarrow{t_{\uparrow 5}}\begin{bmatrix}5\\4\\1\\2\\3\\6\end{bmatrix}\xrightarrow{t_{\uparrow 3}}\begin{bmatrix}1\\5\\4\\2\\3\\6\end{bmatrix}\xrightarrow{t_{\uparrow 3}}\begin{bmatrix}4\\1\\5\\2\\3\\6\end{bmatrix}\xrightarrow{t_{\uparrow 6}}\begin{bmatrix}6\\4\\1\\5\\2\\3\end{bmatrix}\xrightarrow{t_{\uparrow 3}}\begin{bmatrix}1\\6\\4\\5\\2\\3\end{bmatrix}\xrightarrow{t_{\uparrow 3}}\begin{bmatrix}4\\1\\6\\5\\2\\3\end{bmatrix}\xrightarrow{t_{\uparrow 5}}\begin{bmatrix}2\\4\\1\\6\\5\\3\end{bmatrix}\xrightarrow{t_{\uparrow 3}}\begin{bmatrix}1\\2\\4\\6\\5\\3\end{bmatrix}\xrightarrow{t_{\uparrow 3}}\begin{bmatrix}4\\1\\2\\6\\5\\3\end{bmatrix}\xrightarrow{t_{\uparrow 5}}\begin{bmatrix}5\\4\\1\\2\\6\\3\end{bmatrix}\xrightarrow{t_{\uparrow 3}}\begin{bmatrix}1\\5\\4\\2\\6\\3\end{bmatrix}\xrightarrow{t_{\uparrow 3}}$$

Figure 1. An error-correcting rank-modulation Gray code in S_6 with minimal ℓ_∞ distance $d = 3$, presented in more detail later in Example 19.

Gray code with transition set $T = \{t_{i\uparrow j} \mid i < j \leqslant n\}$. When $i = 1$ we refer to C as a "push-to-the-top" code and denote it $G_\uparrow(n, M)$, and we likewise denote "push-to-the-bottom" codes $G_\downarrow(n, M)$.

Example 2 We observe (a fact that has been remarked in [23]) that

$$\begin{bmatrix}1\\2\\3\end{bmatrix}\xrightarrow{t_{\uparrow 2}}\begin{bmatrix}2\\1\\3\end{bmatrix}\xrightarrow{t_{\uparrow 3}}\begin{bmatrix}3\\2\\1\end{bmatrix}\xrightarrow{t_{\uparrow 3}}\begin{bmatrix}1\\3\\2\end{bmatrix}\xrightarrow{t_{\uparrow 2}}\begin{bmatrix}3\\1\\2\end{bmatrix}\xrightarrow{t_{\uparrow 3}}\begin{bmatrix}2\\3\\1\end{bmatrix}\circlearrowright^{t_{\uparrow 3}}$$

is a $G_\uparrow(3, 6)$, i.e., a complete cyclic "push-to-the-top" Gray code over S_3. □

It is worthwhile to note that when S is a group, and T consists of the group action of some subset on S, and C is a (complete- and/or cyclic-) Gray code generated by $t_1, t_2, \dots, t_{M-1}$ $(, t_M)$, then C can be viewed as a simple path (or circuit) in the Cayley graph with generators from T. Moreover for all $\sigma \in S$ we observe that $(\sigma, t_1(\sigma), t_2(t_1(\sigma)), \dots)$ is also a (complete- and/or cyclic- respectively) Gray code. In other words, the code is shift invariant as Cayley graphs are vertex-transitive. In these cases we might refer to the transition sequence generating the code as the code itself, when desirable for simplicity. It is of particular interest to observe that $t_{i\uparrow j}(\sigma) = \sigma \circ (j, j-1, \dots, i)$, i.e., "push-to-the-$i$th-index" transitions are indeed group actions, hence we shall make that simplification in places. We remark that the Cayley graphs generated by $\{(j, j-1, \dots, 1) \mid j \in [n]\}$, i.e., prefix-rotations or "push-to-the-top" operations, were named $(n-1, n)$-Faber–Moore graphs in [16], or (the transpose/converse to) n-rotator graphs in [10]; as mentioned above, we follow the terminology used in more recent works.

When S is equipped with a metric $d_\mathcal{M} : S \times S \to \mathbb{R}_+$, and $C \subseteq S$ has the property that for all $\sigma, \tau \in C$ either $\sigma = \tau$ or $d_\mathcal{M}(\sigma, \tau) \geqslant d$, for some constant $d > 0$, then C (when considered as an unordered set) is commonly referred to as an *error-correcting code* with *minimal distance* d. If $d_\mathcal{M}(\cdot, \cdot)$ models an error mechanism, such that a single error corresponds to distance 1, and $2p + q < d$, it is well known that C can then correct p errors, and also detect q additional errors (e.g., see [32, Prop. 1.5]).

Error-correcting Gray codes were (as mentioned above) named *spread-d circuit codes* (e.g., in [21]), where they were defined by requiring that for all $c_r, c_{r'} \in C$,

$$(r - r' \bmod |C|) \geqslant d \implies d_\mathcal{M}(c_r, c_{r'}) \geqslant d.$$

In that way, e.g., spread-1 circuit codes are simply Gray codes. This eased requirement was made necessary since, working with the Hamming distance d_H in the n-cube, one cannot have codewords at distance less than d in the code sequence attain a distance of at least d. We shall depart from it here to deal with Gray codes which are classic error-correcting codes, but the codes presented in this paper are nevertheless also, in particular, spread-d circuit-codes. This is naturally true in the special case of $d = 2$, which to the authors' knowledge is the only case of error-correcting codes studied thus far in the context of rank modulation with the Gray property. In an analogue to classic Gray codes in the hypercube mentioned above, using the Hamming distance [37], they were dubbed *snake-in-the-box* codes regardless of the metric being used on S_n, although only two such metrics were considered [44].

We shall focus on the ℓ_∞-metric defined on S_n by

$$d_\infty(\sigma, \tau) = \max_{j \in [n]} |\sigma(j) - \tau(j)| \,.$$

That is, it is the metric induced on S_n by the embedding into $\mathbb{Z}^n$ (and, indeed, $\mathbb{R}^n$) implied by the vector notation, and the ℓ_∞-metric in these spaces.

Example 3 In S_4, the code

$$\begin{bmatrix}1\\2\\3\\4\end{bmatrix}\xrightarrow{t_{\uparrow 2}}\begin{bmatrix}2\\1\\3\\4\end{bmatrix}\xrightarrow{t_{\uparrow 3}}\begin{bmatrix}3\\2\\1\\4\end{bmatrix}\xrightarrow{t_{\uparrow 2}}\begin{bmatrix}2\\3\\1\\4\end{bmatrix}\circlearrowright^{t_{\uparrow 3}}$$

has minimal distance 2 (e.g., $d_\infty([1, 2, 3, 4], [2, 3, 1, 4]) = 2$, but as 4 is fixed no two codewords have distance 3). In contrast,

$$\begin{bmatrix}4\\1\\2\\3\end{bmatrix}\xrightarrow{t_{\uparrow 3}}\begin{bmatrix}2\\4\\1\\3\end{bmatrix}\xrightarrow{t_{\uparrow 3}}\begin{bmatrix}1\\2\\4\\3\end{bmatrix}\circlearrowright^{t_{\uparrow 3}}$$

has minimal distance 3 (which we can verify to be the diameter of the metric space (S_4, d_∞)). □

Error-correcting codes in S_n with d_∞ were studied in [39], where they were dubbed *limited-magnitude rank-modulation* codes. A code C with minimal distance d was denoted as an $(n, |C|, d)$-LMRM code. In our case, if a $G_{i\uparrow}(n, M)$ is also an (n, M, d)-LMRM code, we shall denote it a $G_{i\uparrow}(n, M, d)$ (likewise for $G_\uparrow$ and $G_\downarrow$).

III. Auxiliary construction

Before we present the main construction of our paper, we first describe in this section a construction for auxiliary codes which will be a component of the main construction.

TABLE I
Code notations for $C \subseteq S_n$.

Notation	Definition
$G_{i\uparrow}(n, M)$	$C = (c_r)_{r=1}^{M} \subseteq S_n$ such that for all r: $c_{(r \bmod M)+1} = t_{i\uparrow j_r}(c_r)$.
$G_{\uparrow}(n, M)$	C is a $G_{1\uparrow}(n, M)$.
$G_{\downarrow}(n, M)$	$C = (c_r)_{r=1}^{M} \subseteq S_n$ such that for all r: $c_{(r \bmod M)+1} = t_{\downarrow j_r}(c_r)$.
(n, M, d)-LMRM	$C \subseteq S_n$, $\lvert C\rvert = M$ and for all $c_1 \neq c_2 \in C$: $d_\infty(c_1, c_2) \geqslant d$.
$G_{i\uparrow}(n, M, d)$	C is a $G_{i\uparrow}(n, M)$ and an (n, M, d)-LMRM.
$G_{\uparrow}(n, M, d)$	C is a $G_{1\uparrow}(n, M, d)$.
$G_{\uparrow}^{\mathrm{aux}}(k, M)$	C is a $G_{\uparrow}(k, M)$, beginning with $(\mathrm{Id}, t_{\uparrow k}\,\mathrm{Id})$, and for all $q \in [k-1]$: $\sigma \in C \implies (q, k)\sigma \notin C$. (See Section III.)

We say that $C \subseteq S_k$ is j-nontransposing, for some $j \in [k]$, if for all $q \in [k] \setminus \{j\}$ it holds that

$$\sigma \in C \implies (q, j) \circ \sigma \notin C.$$

Unlike some of the codes mentioned thus far, if we shift the first permutation of a j-nontransposing $G_{i\uparrow}(k, M)$ code C, or rotate the generating sequence of transitions, it is no longer assured that the resulting code will be j-nontransposing. We therefore make further requirements and define an *auxiliary code* $G_{\uparrow}^{\mathrm{aux}}(k, M)$ as a $G_{\uparrow}(k, M)$ which is k-nontransposing, beginning at Id, and its first transition is $t_{\uparrow k}$. We will use such codes in our main construction, and we therefore study their existence.

Firstly, note that the only existing $G_{\uparrow}^{\mathrm{aux}}(2, M)$ codes are the singletons $\{\mathrm{Id}\}, \{(1, 2)\}$. However, for $k \geqslant 3$ there do exist $G_{\uparrow}^{\mathrm{aux}}(k, M)$ codes with $M \geqslant 3$, as one such example is

$$\left(\mathrm{Id}, t_{\uparrow 3}\,\mathrm{Id}, {t_{\uparrow 3}}^2\,\mathrm{Id}\right).$$

We also note the following:

Lemma 4 If $C \subseteq S_k$ is k-nontransposing, then $M \leqslant \frac{|S_k|}{2}$.

Proof: Take $q \in [k-1]$, and observe that $\sigma \mapsto (q, k)\sigma$ is an S_k-automorphism, under which C and its image are disjoint. Hence $2M \leqslant |S_k|$. ∎

This motivates us to examine another family of codes, namely, parity-preserving codes, due to the following observations.

Lemma 5 If $C \subseteq S_k$ is parity-preserving then $|C| \leqslant \frac{|S_k|}{2}$.

Proof: Either $C \subseteq A_k$ or $C \subseteq S_k \setminus A_k$. It is well-known that $|A_k| = \frac{|S_k|}{2}$. ∎

Lemma 6 If $C \subseteq S_k$ is a parity-preserving $G_{\uparrow}(k, M)$, then C is k-nontransposing.

Proof: Take $\sigma \in C$ and observe that $\operatorname{sign}\sigma \neq \operatorname{sign}(q, k)\sigma$ for all $q \in [k-1]$, hence $(q, k)\sigma \notin C$, since C is parity-preserving. ∎

Parity-preserving $G_{\uparrow}^{\mathrm{aux}}(2m+1, M)$ codes are known to exist, achieving the aforementioned bound.

Lemma 7 [20] For all $m \neq 2$, there exist parity-preserving $G_{\uparrow}(2m+1, \frac{(2m+1)!}{2})$ codes. The largest parity-preserving $G_{\uparrow}(5, M)$ codes have $M = 57$.

Although not declared, it is shown in [20] that such codes can be assumed to have $t_{\uparrow 2m+1}$ as the first transition in their generating transition sequence, and furthermore, that they also employ at least one $t_{\uparrow 2m-1}$ transition.

In comparison, as noted in [44], a parity-preserving $G_{\uparrow}(2m, M)$ must satisfy $M \leqslant \frac{|S_{2m}|}{2m}$, as it must never employ a $t_{\uparrow 2m}$ transition. This evidently yields much smaller codes than the case of odd orders, and we therefore examine more general $G_{\uparrow}^{\mathrm{aux}}$ codes, which are not parity-preserving. We begin by noting the following lemma.

Lemma 8 [10, Thm. 4] For all $n \geqslant 1$ there exist $G_{\uparrow}(n, n!)$ codes, that is, complete and cyclic "push-to-the-top" Gray codes over the symmetric group S_n.

Relying on these codes, we construct auxiliary codes in the following theorem. The method we apply here of using "push-to-the-bottom" transitions was also used in [10], [23] as a building block for their proposed constructions of a complete and cyclic Gray code in S_n (which are equivalent), then later in [44] for parity-preserving codes in S_{2m+1}.

Theorem 9 For all $m \geqslant 2$ there exists a $G_{\uparrow}^{\mathrm{aux}}(2m, \frac{|S_{2m}|}{2m-1})$.

Proof: Take a $G_{\uparrow}(2m-2, (2m-2)!)$ code C', provided by Lemma 8. We follow the concept of [23, Thm. 7] in extending C' to S_{2m}. Let us define

$$\sigma_0 = t_{\uparrow 2m}\,\mathrm{Id} = [2m, 1, \ldots, 2m-1].$$

If we take $t_{\uparrow i_1}, t_{\uparrow i_2}, \ldots, t_{\uparrow i_{(2m-2)!}}$ to be the transition sequence generating C', then the transition sequence

$$t_{\downarrow 2m+1-i_1}, t_{\downarrow 2m+1-i_2}, \ldots, t_{\downarrow 2m+1-i_{(2k-2)!}}$$

of "push-to-the-bottom" operations, applied in succession to σ_0, generates $C'' \subseteq S_{2m}$, a $G_{\downarrow}(2m, (2m-2)!)$, all of whose elements' vector notations begin with $[2m, 1]$.

We now note that $t_{\downarrow 2m+1-j} = {t_{\uparrow 2m}}^{2m-1} t_{\uparrow j}$. Thus, by replacing each $t_{\downarrow 2m+1-j}$ with $t_{\uparrow j}$ followed by a sequence of $2m-1$ occurrences of $t_{\uparrow 2m}$, we get $C \subseteq S_{2m}$, a $G_{\uparrow}(2m, (2m-2)!2m)$, where every block of $2m$ elements is comprised of cyclic shifts of some $\sigma \in C''$.

The code C is known to be a Gray code [23, Thm. 7]. Moreover, if $\sigma \in C$ satisfies $\tau = (q, 2m)\sigma \in C$, note that both have a vector notation with 1 immediately (cyclically) following $2m$, but since $\tau = (q, 2m)\sigma$ its vector notation has 1 following q. It follows (by abuse of notation) that $q = 2m$. Hence, C is $2m$-nontransposing.

Finally, note that C is generated by a transition sequence ending with $2m-1$ instances of $t_{\uparrow 2m}$, and begins with $\sigma_0 = t_{\uparrow 2m}\,\mathrm{Id}$. Therefore, it includes Id followed by a $t_{\uparrow 2m}$ transition. A cyclic shift of C therefore satisfies the theorem. ∎

Example 10 To construct a $G_{\uparrow}^{\mathrm{aux}}(4, 8)$ we utilize the complete $G_{\uparrow}(2, 2)$ code $\{\mathrm{Id}, t_{\uparrow 2}\,\mathrm{Id}\}$, generated by $t_{\uparrow 2}, t_{\uparrow 2}$, to arrive by the $G_{\downarrow}(4, 2)$ code starting with $\sigma_0 = t_{\uparrow 4}\,\mathrm{Id} = [4, 1, 2, 3]$:

$$C'' = \{[4, 1, 2, 3], [4, 1, 3, 2]\},$$

which is generated by $t_{\downarrow 3}, t_{\downarrow 3}$. We recall that $t_{\downarrow 3} = t_{\uparrow 4}{}^{3} \circ t_{\uparrow 3}$, allowing us to expand C'' in the following manner:

$$\begin{array}{ccccccccccccccc} \begin{bmatrix}4\\1\\2\\3\end{bmatrix} & \xrightarrow{t_{\uparrow 3}} & \begin{bmatrix}2\\4\\1\\3\end{bmatrix} & \xrightarrow{t_{\uparrow 4}} & \begin{bmatrix}3\\2\\4\\1\end{bmatrix} & \xrightarrow{t_{\uparrow 4}} & \begin{bmatrix}1\\3\\2\\4\end{bmatrix} & \xrightarrow{t_{\uparrow 4}} & \begin{bmatrix}4\\1\\3\\2\end{bmatrix} & \xrightarrow{t_{\uparrow 3}} & \begin{bmatrix}3\\4\\1\\2\end{bmatrix} & \xrightarrow{t_{\uparrow 4}} & \begin{bmatrix}2\\3\\4\\1\end{bmatrix} & \xrightarrow{t_{\uparrow 4}} & \begin{bmatrix}1\\2\\3\\4\end{bmatrix} \circlearrowright t_{\uparrow 4} \\ \Cap & & & & & & & & \Cap & & & & & & \\ C'' & & & & & & & & C'' & & & & & & \end{array}$$

Finally, we observe that as seen in the proof to Theorem 9, the code we constructed has Id as its last codeword, followed by a $t_{\uparrow 4}$ operation. A cyclic shift of the code now begins with Id and the required operation, which satisfies Theorem 9. $\square$

We remark that, while Theorem 9 does not produce codes much larger than the parity-preserving code of size $\frac{|S_{2m}|}{2m}$, it does at least allow us to permute the last element $2m$ while preserving the property of being $2m$-nontransposing, and thus construct auxiliary codes.

Next, we present another construction which yields larger codes, for even $k \geqslant 6$ (but not $k = 4$). From now on, we fix $m \geqslant 2$ and let $k = 2m + 2$. We also define $\varphi : S_{2m+2} \longrightarrow S_{2m+2}$ by

$$\varphi = t_{\uparrow 2m+2}{}^{2} \circ t_{\uparrow 2m-1}{}^{-1}.$$

We note that

$$\varphi(\pi) = \pi \circ (1, 2m + 1)(2m + 2, 2m, 2m - 1, \ldots, 2),$$

Hence, informally, in π's vector notation, φ transposes the elements in indices $1, 2m + 1$, and cyclically shifts all other elements once to the bottom (i.e., as if applying a "push-to-the-top" operation on the last index – acting only on these indices). We can also observe that $\varphi^{2m} = \mathrm{Id}$.

We conveniently define, for all $r \geqslant 0$, the permutations

$$\begin{aligned} \hat{\pi}_r &= \varphi^{r}(\mathrm{Id}) \\ &= (1, 2m + 1)^{r}(2m + 2, 2m, 2m - 1, \ldots, 2)^{r} \in S_{2m+2}, \end{aligned}$$

In particular, we note that when $r \equiv r' \pmod{2m}$, and only then, we have $\hat{\pi}_r = \hat{\pi}_{r'}$.

Lemma 11 For all $r \geqslant 0$ a parity-preserving $G_{\uparrow}(2m + 2, M_{2m+2})$ code P_r exists which begins with $\hat{\pi}_r$ and ends with $t_{\uparrow 2m-1}{}^{-1}\hat{\pi}_r$, where

$$M_{2m+2} = \begin{cases} 57 & m = 2, \\ \frac{(2m+1)!}{2} & m > 2. \end{cases}$$

Proof: The claim follows trivially from the codes provided by Lemma 7, if we shift the generating transition sequence such that it ends with $t_{\uparrow 2m-1}$ and apply it to $\hat{\pi}_r$. $\blacksquare$

We note in particular that for all r, $\hat{\pi}_r$ is even, and thus $P_r \subseteq A_{2m+2}$. Moreover, since the parity-preserving code P_r does not employ $t_{\uparrow 2m+2}$, for all $\pi \in P_r$ it holds that

$$\begin{aligned} \pi(2m + 2) &= \hat{\pi}_r(2m + 2) \\ &= \begin{cases} 2m + 2 & r \equiv 0 \pmod{2m}, \\ 2m + 1 - (r \bmod 2m) & r \not\equiv 0 \pmod{2m}. \end{cases} \end{aligned}$$

Thus, when considered as sets,

$$P_r \cap P_{r'} = \emptyset,$$

for all $0 \leqslant r < r' < 2m$.

We shall construct a $G_{\uparrow}^{\mathrm{aux}}(2m + 2, M)$ code by stitching together $P_1, P_2, \ldots, P_{2m-1}$. We will need to amend P_0 before incorporating it into our code, for reasons we shall discuss below. First, we describe the stitching method in the following lemma.

Lemma 12 For all $r \geqslant 0$ (including, in particular, $r = 2m - 1$), we may concatenate P_r, P_{r+1} into a (non-cyclic) "push-to-the-top" code by applying the transitions $t_{\uparrow 2m+2}, t_{\uparrow 2m+2}$ to the last permutation of P_r, which is $t_{\uparrow 2m-1}{}^{-1}\hat{\pi}_r$. Additionally, the only odd permutation in the resulting code is

$$\beta_{r+1} = t_{\uparrow 2m+2}{}^{-1}(\hat{\pi}_{r+1}).$$

We shall refer to it as the $(r + 1)$*-bridge*.

Proof: The claim follows trivially from the definition

$$\hat{\pi}_{r+1} = \varphi(\hat{\pi}_r) = t_{\uparrow 2m+2} \circ \left(t_{\uparrow 2m+2} \circ t_{\uparrow 2m-1}{}^{-1}(\hat{\pi}_r)\right),$$

since P_r, P_{r+1} are parity-preserving, and $t_{\uparrow 2m+2}$ flips parity. $\blacksquare$

Lemma 12 can be used iteratively to concatenate $P_1, P_2, \ldots, P_{2m-1}$, with a single odd permutation–the r-bridge–between each pair of P_{r-1}, P_r. Thus, we obtain the sequence

$$P_1, \beta_2, P_2, \beta_3, \ldots, \beta_{2m-1}, P_{2m-1}.$$

Note that if any two permutations π_1, π_2 in the resulting sequence satisfy $\pi_1 = (q, 2m + 2) \circ \pi_2$ for some $q \in [2m + 1]$, then w.l.o.g π_2 is odd and hence an r-bridge for some r, and π_1 is even and thus not a bridge. Since in every bridge the last element is

$$\beta_r(2m + 2) = \hat{\pi}_r(1) \in \{1, 2m + 1\},$$

and in every non-bridge it is not, it must follow, then, that $q = \beta_r(2m + 2)$, and in particular

$$\begin{aligned} \pi_1(2m + 2) &= (\beta_r(2m + 2), 2m + 2) \circ \beta_r(2m + 2) \\ &= 2m + 2, \end{aligned}$$

thus $\pi_1 \in P_0$.

We witness, therefore, that no such pair of permutations exist, since we have not yet incorporated P_0 into our code. Hence, so far we have a $(2m + 2)$-nontransposing code. It also becomes apparent that P_0 must necessarily be amended prior to its inclusion, so it does not include any permutations of the form

$$(\beta_r(2m + 2), 2m + 2) \circ \beta_r, \qquad 0 < r \leqslant 2m.$$

In order to do so, we note that for all $r \geqslant 0$

$$\beta_r(2m) = \hat{\pi}_r(2m + 1) \in \{1, 2m + 1\},$$

and in particular $\beta_r(2m) \neq 2m + 2$, hence

$$(\beta_r(2m + 2), 2m + 2) \circ \beta_r(2m) = \beta_r(2m) \in \{1, 2m + 1\}.$$

It follows that if we let P'_0 be generated by the transition sequence $t_{\uparrow 2m-1}{}^{2m-1}$ applied to $\hat{\pi}_0$, then it is parity-preserving, its last permutation is $t_{\uparrow 2m-1}{}^{-1}\hat{\pi}_0$, and for all $\pi \in P'_0$ we have $\pi(2m) = 2m \notin \{1, 2m + 1\}$, thus

$$P'_0 \cap \{(\beta_r(2m + 2), 2m + 2) \circ \beta_r\}_{r=1}^{2m} = \emptyset.$$

Lemma 13 The following sequence P,

$$P = P'_0, \beta_1, P_1, \beta_2, P_2, \beta_3, \dots, \beta_{2m-1}, P_{2m-1}, \beta_{2m},$$

is a cyclic and k-nontransposing $G_{\uparrow}(k, M)$.

Proof: By Lemma 12, and since when considered as sets,

$$P_r \cap P_{r'} = \emptyset$$

for all $0 < r < r' < 2m$, and similarly P'_0 is disjoint from $P_1, P_2, \dots, P_{2m-1}$, we know that P is a $G_{\uparrow}(2m+2, M)$.

As seen above, if for any two permutations $\pi_1, \pi_2 \in P$ and $q \in [2m+1]$ we have $\pi_1 = (q, 2m+2) \circ \pi_2$, then w.l.o.g. $\pi_2 = \beta_r$ for some $0 < r \leqslant 2m$ and $q = \beta_r(2m+2)$. In particular, $\pi_1(2m+2) = 2m+2$, thus

$$\pi_1 \notin \{\beta_r\}_{r=1}^{2m} \cup \bigcup_{r=1}^{2m-1} P_r,$$

thus $\pi_1 \in P'_0$. But

$$P'_0 \cap \{(\beta_r(2m+2), 2m+2) \circ \beta_r\}_{r=1}^{2m} = \emptyset,$$

in contradiction. ■

The code from Lemma 13 is almost what we need. The only property lacking is the fact that Id is not followed in P by the transition $t_{\uparrow 2m+2}$. We fix this in the following theorem.

Theorem 14 Let $k \geqslant 6$ be even. Then there exists a $G_{\uparrow}^{\text{aux}}(k, M)$, with

$$M = \begin{cases} 178 & k = 6, \\ (k-3)\left(\frac{(k-1)!}{2} + 2\right) + 1 & k > 6. \end{cases}$$

In particular, for all $k > 6$,

$$M > \frac{k-3}{k} \cdot \frac{k!}{2}.$$

Proof: Denote $k = 2m+2$ for $m \geqslant 2$, and let $P = (c_j)_{j=1}^{M}$ be the code from Lemma 13. Since $\mathrm{Id} \in S_k$ is not followed with a $t_{\uparrow k}$ transition in P, we denote the last permutation of P'_0 by $\tilde{\pi}$, and replace P with

$$\tilde{P} = \tilde{\pi}^{-1} P = \left(\tilde{\pi}^{-1} \circ c_j\right)_{j=1}^{M}.$$

We observe that $\tilde{P}$ is still a "push-to-the-top" code since "push-to-the-top" transitions are group actions by right-multiplications. Moreover, since $\tilde{\pi}(k) = k$, if for some $\pi_1, \pi_2 \in P$ we have $\tilde{\pi}^{-1} \circ \pi_1 = (q, k) \circ \left(\tilde{\pi}^{-1} \circ \pi_2\right)$, where $q \in [k-1]$, then

$$\begin{aligned} \pi_1 &= \tilde{\pi} \circ \left[(q, k) \circ \left(\tilde{\pi}^{-1} \circ \pi_2\right)\right] \\ &= \left[\tilde{\pi} \circ (q, k) \circ \tilde{\pi}^{-1}\right] \circ \pi_2 = (\tilde{\pi}(q), k) \circ \pi_2, \end{aligned}$$

and $\tilde{\pi}(q) \in [k-1]$, in contradiction.

As for the size of the code, note that $\left|P'_0\right| = 2m-1 = k-3$ and

$$|P_1| = |P_2| = \dots = |P_{2m-1}| = \begin{cases} 57 & k = 6, \\ \frac{(k-1)!}{2} & k > 6. \end{cases}$$

Counting $\beta_1, \dots, \beta_{2m}$, the claim is thus substantiated, up to a rotation to make Id the first permutation. ■

To conclude this section, we combine Lemma 7, Theorem 9 and Theorem 14 into the following corollary.

Corollary 15 For all $k \geqslant 3$ there exists a $G_{\uparrow}^{\text{aux}}(k, \tilde{M}_k)$, where

$$\tilde{M}_k = \begin{cases} 8 & k = 4 \\ 57 & k = 5 \\ 178 & k = 6 \\ \frac{k!}{2} & 5 \neq k \equiv 1 \pmod 2 \\ \rho_k \frac{k!}{2} & 6 < k \equiv 0 \pmod 2, \end{cases}$$

and $\rho_k > \frac{k-3}{k}$.

IV. Code Construction

In this section we present the main construction of our paper, and discuss the size and asymptotic rate of the resulting codes. We will show, surprisingly, that our method generates codes which are larger than formerly known families of codes, even though we require the additional structure of a Gray code.

A. Main code construction

We now present a construction of $G_{\uparrow}(n, M, d)$ codes, for $d < n$, which we base on Corollary 15 and Lemma 8.

It will simplify the presentation to assume $n = kd$ for some positive $k \geqslant 2$, since in that case every congruence class modulo d of $[n]$ has size k. Nonetheless, the construction is applicable to any $n > d$ with natural amendments. We discuss these changes, focusing on special cases, after presenting the simple construction first.

Construction A Let $n, k, d \in \mathbb{N}$, with $n = kd$ and $k \geqslant 2$. We recursively construct a sequence of codes, $C_d, C_{d-1}, \dots, C_1$. An explicit construction is given for C_d and a recursion step constructs C_m from C_{m+1}.

Recursion base: We construct the code C_d by starting at the permutation $\sigma_0 \in S_n$ defined by

$$\sigma_0(j) = d\,(j \bmod k) + \left\lceil \frac{j}{k} \right\rceil.$$

We obtain a transition sequence $t_{\uparrow r_1}, t_{\uparrow r_2}, \dots, t_{\uparrow r_{k!}}$ which generates the $G_{\uparrow}(k, k!)$ provided by Lemma 8. The code C_d starts with σ_0, and uses the transition sequence

$$t_{k(d-1)+1 \uparrow k(d-1)+r_1}, t_{k(d-1)+1 \uparrow k(d-1)+r_2}, \dots$$
$$\dots, t_{k(d-1)+1 \uparrow k(d-1)+r_{k!}}.$$

Recursion step: Assume C_{m+1} has already been constructed, starting with permutation σ_0. Additionally, let

$$t_{\uparrow k+1}, t_{\uparrow i_2}, \dots, t_{\uparrow i_{\tilde{M}_{k+1}}} \tag{1}$$

be a transition sequence generating a $G_{\uparrow}^{\text{aux}}(k+1, \tilde{M}_{k+1})$ code provided by Corollary 15.

We construct the code C_m as follows: replace each $t_{km+1 \uparrow j}$ transition of C_{m+1} with $t_{k(m-1)+1 \uparrow j}$, followed by $t_{k(m-1)+1\uparrow k(m-1)+i_2}$, $t_{k(m-1)+1\uparrow k(m-1)+i_3}$, and so on until $t_{k(m-1)+1\uparrow k(m-1)+i_{\tilde{M}_{k+1}}}$. □

Lemma 16 For all $n = kd$, $k \geqslant 2$, the code C_d from Construction A is a $G_{k(d-1)+1\uparrow}(n, k!, d)$.

Proof: The parameters of the code are obvious, except perhaps the minimal distance d. The fact that the codewords of C_d are distinct follows from Lemma 8.

To prove the minimal distance d, note that for all $0 \leqslant u < d$ and $ku + 1 \leqslant i < j \leqslant k(u + 1)$ it holds that $\sigma_0(i) \equiv \sigma_0(j) \pmod{d}$. Thus, for every distinct $\sigma, \tau \in C_d$, there exists j, $k(d-1) < j \leqslant kd = n$, such that $\sigma(j) \neq \tau(j)$. Since by construction $\sigma(j) \equiv \tau(j) \equiv 0 \pmod{d}$, we observe

$$d_\infty(\sigma, \tau) \geqslant |\sigma(j) - \tau(j)| \geqslant d,$$

implying that C_d is a $G_{k(d-1)+1\uparrow}(n, k!, d)$. ■

Example 17 We let $d = 3$, $k = 2$, and $n = kd = 6$. We construct the code C_3 starting at

$$\sigma_0 = [4, 1, 5, 2, 6, 3] \in S_6.$$

We use the complete $G_\uparrow(2, 2)$ shown in Example 10, which is generated by the sequence $t_{\uparrow 2}, t_{\uparrow 2}$. We arrive at a generating sequence $t_{5\uparrow 6}, t_{5\uparrow 6}$ for C_3. Hence, in our example

$$C_3 = ([4, 1, 5, 2, 6, 3], [4, 1, 5, 2, 3, 6]),$$

which is readily seen to be a $G_{5\uparrow}(6, 2, 3)$ code. □

We shall follow Construction A to develop this example into a $G_\uparrow(6, 18, 3)$ in Example 19. First, we prove the validity of the construction.

Theorem 18 For all $n = kd$, $k \geqslant 2$, the code C_1 from Construction A is a $G_\uparrow(n, \tilde{M}_{k+1}{}^{d-1} \cdot k!, d)$.

Proof: To prove the claim we will prove by induction that C_m from Construction A, for all $m \in [d]$, is a $G_{k(m-1)+1\uparrow}(n, \tilde{M}_{k+1}{}^{d-m} \cdot k!, d)$. The base case of C_d was proved in Lemma 16. Assume the claim holds for C_{m+1} and we now prove it for C_m.

Recall (1) gives the sequence of transitions for a $G_\uparrow^{\text{aux}}(k + 1, \tilde{M}_{k+1})$. Then

$$t_{\uparrow i_{\tilde{M}_{k+1}}} t_{\uparrow i_{\tilde{M}_{k+1}-1}} \cdots t_{\uparrow i_3} t_{\uparrow i_2} = t_{\uparrow k+1}{}^{-1}.$$

Thus,

$$t_{km+1\uparrow j} = \left(\prod_{r=2}^{\tilde{M}_{k+1}} t_{k(m-1)+1\uparrow k(m-1)+i_r} \right) t_{k(m-1)+1\uparrow j}$$

(where the product is expanded right-to-left). Therefore, C_m expands each "push-to-the-$(km + 1)$st-index" transition of C_{m+1} into $\tilde{M}_{k+1}$ "push-to-the-$[k(m-1) + 1]$st-index" transitions.

It follows that C_m contains the codewords of C_{m+1} in the same order, with $\tilde{M}_{k+1} - 1$ new words inserted between any two words originally from C_{m+1}. We say that each codeword of C_{m+1} (now appearing in C_m) is the C_{m+1}-*parent* of each of the $\tilde{M}_{k+1}$ preceding codewords in C_m (including itself), since their vector notations agree on the order of the elements

$$\sigma_0(km + 1), \sigma_0(km + 2), \ldots, \sigma_0(n).$$

We note here (and will further examine later) that when $x = \sigma'(km + 1)$ and σ' is a C_{m+1}-parent of $\tilde{M}_{k+1}$ codewords in C_m (inclusive), then that subsequence of C_m is an x-nontransposing $G_{k(m-1)+1\uparrow}(n, \tilde{M}_{k+1})$ code. This follows since we used an auxiliary code to construct that subsequence, of which the parent takes the role of first permutation, and in σ', x is the last element among the indices being permuted.

Now, suppose that $\sigma, \tau \in C_m$ satisfy $d_\infty(\sigma, \tau) < d$. Let σ', τ' be their C_{m+1}-parents, respectively. To complete the proof we will show that $\sigma = \tau$.

Case 1: $\sigma' = \tau'$. Denote

$$x = \sigma'(km + 1) = \tau'(km + 1)$$

and $s = \sigma^{-1}(x)$, $a = \tau(s)$.

If $a = x$ then for all $j \neq s$, $k(m-1) < j \leqslant km + 1$, we have $\sigma(j) \equiv \tau(j) \pmod{d}$ and

$$|\sigma(j) - \tau(j)| \leqslant d_\infty(\sigma, \tau) < d,$$

hence $\sigma(j) = \tau(j)$, and $\sigma = \tau$.

Otherwise, $a \neq x$, and denote $t = \tau^{-1}(x) \neq s$. It similarly holds for all $j \notin \{s, t\}$, $k(m-1) < j \leqslant km + 1$, that $\sigma(j) = \tau(j)$. We therefore observe $\tau = \sigma \circ (s, t)$. This implies that, if we let $\hat{\sigma}, \hat{\tau} \in S_{k+1}$ be the permutations in the $G_\uparrow^{\text{aux}}$ we obtained, generated similarly to σ, τ, respectively (i.e., by their corresponding transition sequences), then

$$\hat{\tau} = \hat{\sigma} \circ (s - k(m-1), t - k(m-1)) = (q, k + 1)\hat{\sigma}$$

for some $q \in [k]$, in contradiction to the fact it was a $G_\uparrow^{\text{aux}}(k + 1, \tilde{M}_{k+1})$.

Case 2: $\sigma' \neq \tau'$. Since $\sigma', \tau' \in C_{m+1}$ we have by assumption $d_\infty(\sigma', \tau') \geqslant d$, and note that for all j satisfying $j \leqslant k(m-1)$ or $j > km + 1$, it holds that $\sigma(j) = \sigma'(j)$ and $\tau(j) = \tau'(j)$. Hence there exists j, $k(m-1) < j \leqslant km + 1$, such that

$$|\sigma(j) - \tau(j)| < d \quad \text{but} \quad |\sigma'(j) - \tau'(j)| \geqslant d.$$

Note particularly, since for all $k(m-1) < j \leqslant km$ it holds that $\sigma'(j) = \sigma_0(j) = \tau'(j)$, that we have

$$|\sigma'(km + 1) - \tau'(km + 1)| \geqslant d.$$

Denote $x = \sigma'(km + 1)$, $y = \tau'(km + 1)$, and note that

$$\{\sigma(j)\}_{j=k(m-1)+1}^{km+1} = \{a_i\}_{i=1}^{k} \cup \{x\};$$
$$\{\tau(j)\}_{j=k(m-1)+1}^{km+1} = \{a_i\}_{i=1}^{k} \cup \{y\},$$

where $\{a_i\}_{i=1}^{k}$ is a congruence class modulo d of $[n]$, of which x, y are not members.

Let $s = \sigma^{-1}(x)$ and denote $a = \tau(s)$. Since

$$|x - a| = |\sigma(s) - \tau(s)| \leqslant d_\infty(\sigma, \tau) < d$$

we have $a \neq y$. Let $t = \sigma^{-1}(a)$. Since $a \in \{a_i\}_{i=1}^{k}$ is a congruence class modulo d, for all $b \in \{a_i\}_{i=1}^{k} \setminus \{a\}$ we observe $|a - b| \geqslant d$, but

$$|a - \tau(t)| = |\sigma(t) - \tau(t)| \leqslant d_\infty(\sigma, \tau) < d$$

and therefore $\tau(t) = y$. For all $j \notin \{s,t\}$ satisfying $k(m-1) < j \leqslant km+1$ we then have $\sigma(j) \equiv \tau(j) \pmod d$ and $|\sigma(j)-\tau(j)| \leqslant d_\infty(\sigma,\tau) < d$, hence $\sigma(j) = \tau(j)$.

This implies that, if we again let $\hat{\sigma}, \hat{\tau} \in S_{k+1}$ be the permutations in the $G_{\uparrow}^{\text{aux}}$ generated similarly to σ, τ respectively, then

$$\hat{\tau} = \hat{\sigma} \circ (s-k(m-1), t-k(m-1)) = (q, k+1)\hat{\sigma}$$

where q is given by $a = a_q \in \{a_i\}_{i=1}^{k}$, again contradicting the properties of a $G_{\uparrow}^{\text{aux}}(k+1, \tilde{M}_{k+1})$. Hence C_m has minimal ℓ_∞-distance of at least d, as required. ■

Example 19 We complete Example 17 into a $G_{\uparrow}(6, 3^2 \cdot 2, 3)$ code by applying the recursion step twice. In each step, since $k = 2$, we utilize the trivial parity-preserving $G_{\uparrow}^{\text{aux}}(3,3)$ code generated by the sequence $t_{\uparrow 3}, t_{\uparrow 3}, t_{\uparrow 3}$.

Firstly, recall that we used

$$\sigma_0 = [4,1,5,2,6,3] \in S_6,$$

and the sequence $t_{5\uparrow 6}, t_{5\uparrow 6}$ generates

$$C_3 = ([4,1,5,2,6,3], [4,1,5,2,3,6]).$$

We build C_2 by exchanging each $t_{5\uparrow 6}$ transition by $t_{3\uparrow 6}$ followed by 2 instances of $t_{2+1\uparrow 2+3} = t_{3\uparrow 5}$; the middle level of Figure 2 shows the resulting code.

Secondly, as seen in the same figure, each $t_{3\uparrow j}$ transition of C_2, $j \in \{5,6\}$, can be replaced by $t_{1\uparrow j} = t_{\uparrow j}$, followed by 2 instances of $t_{0+1\uparrow 0+3} = t_{\uparrow 3}$, to generate C_1.

Note that $C_3 \subseteq C_2 \subseteq C_1$, and that they are $G_{5\uparrow}(6,2,3)$, $G_{3\uparrow}(6,6,3)$ and $G_{\uparrow}(6,18,3)$ codes, respectively. □

We now describe the changes needed in Construction A to allow general n and d parameters. We first consider n not necessarily being a multiple of d, but still $n \geqslant 2d$. For all $i \in [d]$, let

$$\mathcal{R}_i = \{i, i+d, i+2d, \dots, n-((n-i) \bmod d)\},$$

be the ith congruence class modulo d of $[n]$. Then

$$|\mathcal{R}_i| = \begin{cases} \left\lceil \frac{n}{d} \right\rceil & 1 \leqslant i \leqslant (n \bmod d), \\ \left\lfloor \frac{n}{d} \right\rfloor & (n \bmod d) < i \leqslant d. \end{cases}$$

We define the starting permutation

$$\sigma_0 = [\mathcal{R}_1 | \mathcal{R}_2 | \dots | \mathcal{R}_d] \in S_n,$$

to be comprised of a concatenation of the congruence classes, where the order of elements within the congruence class is arbitrary. Additionally, the recursion base uses a $G_{\uparrow}(|\mathcal{R}_d|, |\mathcal{R}_d|!)$. As for the recursion step of constructing C_m from C_{m+1}, we can still apply it with the following changes:

- We choose $G_{\uparrow}^{\text{aux}}(|\mathcal{R}_m|+1, \tilde{M}_{|\mathcal{R}_m|+1})$.
- We use push operations to position $1 + \sum_{i=1}^{m-1} |\mathcal{R}_i|$.

We obtain C_1 which is a $G_{\uparrow}(n, M, d)$, where

$$M = \tilde{M}_{\lceil n/d \rceil + 1}{}^{n \bmod d} \cdot \tilde{M}_{\lfloor n/d \rfloor + 1}{}^{d-(n \bmod d)-1} \cdot \left\lfloor \frac{n}{d} \right\rfloor !.$$

Finally, we discuss the special case of $n < 2d$, in which all but $(n \bmod d)$ congruence classes are singletons. We will amend our construction by replacing the recursion base with

$$C_m = \left\{\sigma_0, t_{2m-1\uparrow 2m+1}\sigma_0, t_{2m-1\uparrow 2m+1}{}^2\sigma_0\right\},$$

where $m = n \bmod d$, and continuing the recursion step as discussed above. Thus, we are effectively only using the first member of $\mathcal{R}_{m+1}$ together with the previous congruence classes, fixing $\sigma_0(j)$ for $j > 2m+1$. In this case, we obtain C_1 which is a $G_{\uparrow}(n, 3^{n \bmod d}, d)$.

Thus, in what follows, whenever we mention Construction A, we refer to its most general version applying to all n and d.

B. Code-size analysis and comparison

We would like to give an explicit expression for the size of the codes constructed by Construction A. This would enable a comparison with previously known results.

Lemma 20 Let C_1 be the code from (the general version of) Construction A. Then its size, $|C_1|$, is given by (2).

Proof: Let us first assume $n \geqslant 2d$. We note the asymmetry in Construction A between congruence classes $\mathcal{R}_i$ of odd and even sizes. Indeed, a class of size $|\mathcal{R}_i| = k \geqslant 2$ (for all classes other than $\mathcal{R}_d$, which is used in the recursion base and whose contribution is based on the $G_{\uparrow}(k, k!)$ code) contributes to the code size, according to Corollary 15, a multiplicative factor of

$$\tilde{M}_{k+1} = \begin{cases} 8 & k = 3; \\ 57 & k = 4; \\ 178 & k = 5; \\ \frac{(k+1)!}{2} & 4 \neq k \equiv 0 \pmod 2; \\ \rho_{k+1}\frac{(k+1)!}{2} & 5 < k \equiv 1 \pmod 2, \end{cases}$$

where, again, $\rho_{k+1} > \frac{k-2}{k+1}$.

It is therefore important to note that when $\left\lfloor \frac{n}{d} \right\rfloor \equiv 0 \pmod 2$, $[n]$ has $(n \bmod d)$ congruence classes modulo d of odd size $\left\lceil \frac{n}{d} \right\rceil$, and $d - (n \bmod d)$ classes of even size $\left\lfloor \frac{n}{d} \right\rfloor$. Thus, if additionally $\left\lfloor \frac{n}{d} \right\rfloor > 4$, the constructed code C_1 is of size

$$|C_1| = \left(\rho_{\lceil n/d \rceil + 1}\frac{(\lceil n/d \rceil + 1)!}{2}\right)^{n \bmod d} \cdot \left\lfloor \frac{n}{d} \right\rfloor ! \cdot \left(\frac{(\lfloor n/d \rfloor + 1)!}{2}\right)^{d-(n \bmod d)-1},$$

and simple rearranging gives us the first case of (2). Similar considerations give us the next five cases of (2).

Finally, we consider the case of $n < 2d$, which implies $\left\lfloor \frac{n}{d} \right\rfloor = 1$. In this special case we only permute $(n \bmod d) = (n-d)$ congruence classes of $[n]$, (and each such class has $2 = \left\lfloor \frac{n}{d} \right\rfloor + 1$ elements). As mentioned, we therefore construct a code of size $|C_1| = 3^{n \bmod d}$. ■

We comment that it is also possible to achieve a slight gain in code size by reordering σ_0 so that the last block consists of a congruence class of odd size, rather than even, where the added complexity of index calculation is inconsequential. The gains are negligible for large enough n.

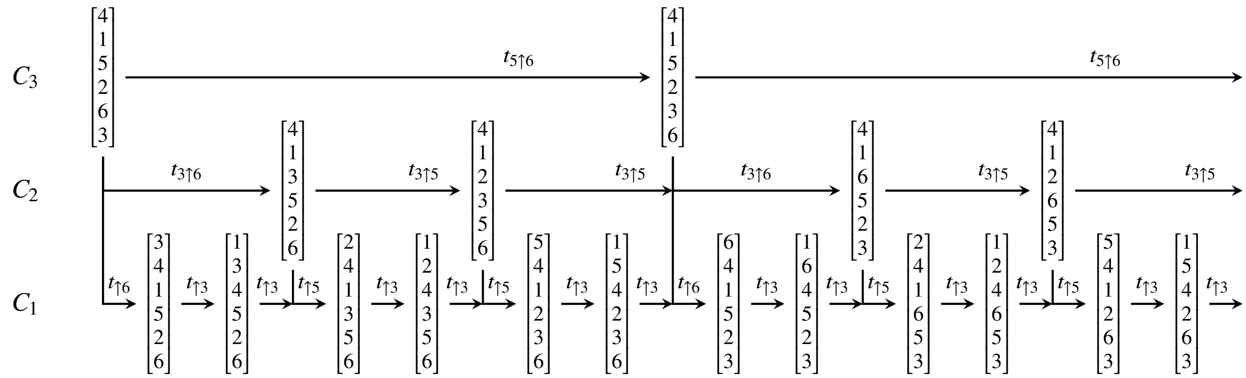

Figure 2. Construction A as demonstrated in the case $d = 3$, $k = 2$.

$$|C_1| = \begin{cases} \left(\left\lceil \frac{n}{d}\right\rceil + 1\right)^{n \bmod d} \left(\left\lfloor \frac{n}{d}\right\rfloor + 1\right)!^{\,d} \cdot \frac{\rho_{\lceil n/d\rceil+1}{}^{n \bmod d}}{2^{d-1}(\lfloor n/d\rfloor+1)} & 4 < \left\lfloor \frac{n}{d}\right\rfloor \equiv 0 \pmod 2, \\ \left(\frac{178}{57}\right)^{n \bmod d} \cdot 57^{d-1} \cdot 24 & \left\lfloor \frac{n}{d}\right\rfloor = 4, \\ \left(\frac{8}{3}\right)^{n \bmod d} \cdot 3^{d-1} \cdot 2 & \left\lfloor \frac{n}{d}\right\rfloor = 2, \\ \left(\left\lceil \frac{n}{d}\right\rceil + 1\right)^{n \bmod d} \left(\left\lfloor \frac{n}{d}\right\rfloor + 1\right)!^{\,d} \cdot \frac{\rho_{\lfloor n/d\rfloor+1}{}^{(d-1)-(n \bmod d)}}{2^{d-1}(\lfloor n/d\rfloor+1)} & 5 < \left\lfloor \frac{n}{d}\right\rfloor \equiv 1 \pmod 2, \\ \left(\frac{1260}{89}\right)^{n \bmod d} \cdot (178)^{d} \cdot \frac{120}{178} & \left\lfloor \frac{n}{d}\right\rfloor = 5, \\ \left(\frac{57}{8}\right)^{n \bmod d} \cdot 8^{d} \cdot \frac{3}{4} & \left\lfloor \frac{n}{d}\right\rfloor = 3, \\ 3^{n \bmod d} & \left\lfloor \frac{n}{d}\right\rfloor = 1. \end{cases} \tag{2}$$

We now turn to comparing the size of the resulting code with that of previously constructed codes, as well as known bounds on the cardinality of such codes.

The first comparison we make is with codes that have the Gray property. Such codes were only studied for $d = 2$, i.e., snake-in-the-box codes or $G_\uparrow(n, M, 2)$ codes in our notation. These codes were studied in [44, Thm. 24], where it was shown that such codes can be constructed with sizes

$$M = \left\lceil \frac{n}{2}\right\rceil! \left(\left\lfloor \frac{n}{2}\right\rfloor + \left(\left\lfloor \frac{n}{2}\right\rfloor - 1\right)!\right).$$

Construction A improves this size by a factor of $\frac{1}{2}\left(\left\lceil \frac{n}{2}\right\rceil + 1\right)\left\lfloor \frac{n}{2}\right\rfloor$, times a factor of $\rho_{\lfloor n/2\rfloor+1}$ when $n \equiv 2 \pmod 4$ (in the case of $n \equiv 1 \pmod 4$, the factor $\rho_{\lfloor n/2\rfloor+1}$ is eliminated by changing the order of congruence classes in σ_0). We note that a similar improvement was made concurrently by [42] in a preprint devoted solely to the case of $d = 2$, i.e., snake-in-the-box codes.

We now also compare our results to error-correcting codes with the ℓ_∞-metric which are not necessarily Gray codes (LMRM-codes). We observe that the best known general LMRM-code construction to date, [39, Cst. 1, Thm. 2] and [27, Sec. III-A], presented (n, M, d)-LMRM codes with sizes

$$M = \left\lceil \frac{n}{d}\right\rceil!^{\,n \bmod d} \left\lfloor \frac{n}{d}\right\rfloor!^{\,d-(n \bmod d)},$$

which our construction improves upon, more pronouncedly the more $[n]$ has even-sized congruence classes modulo d (cf. (2)).

Finally, we also note the following lemma:

Lemma 21 [39, Thm. 16] If $C \subseteq S_n$ is a code with minimal ℓ_∞ distance d, then

$$|C| \leqslant \frac{n!}{(d!)^{\lfloor n/d\rfloor}(n \bmod d)!}$$

We remark that in the case of $d = 2$, [44] confirmed that the optimal size of error-correcting codes for $n = 4, 5, 6$ to be $6, 30, 90$ respectively, meeting the bound of Lemma 21. It also presented Gray codes achieving these sizes by computer search. Searches in higher dimension were reported unfeasible. For higher values of d, the optimal size is unknown, as well as whether Gray codes can achieve it. While the reader can appreciate that the bound of Lemma 21 is exponentially greater than the size provided by Lemma 20, we note anecdotally that the $G_\uparrow(6, 18, 3)$ code presented in Example 19 almost meets the bound ($M \leqslant 20$).

In the asymptotic regime, we go on to examine the case of $d = \Theta(n)$. For an (n, M, d)-LMRM code (and in particular a $G_\uparrow(n, M, d)$), we follow the convention (e.g., [39]) of defining the *rate* of the code

$$R = \frac{1}{n}\log_2 M,$$

and its *normalized distance*

$$\delta = \frac{d}{n}.$$

The following were proven in [39].

Lemma 22 [39, Thm. 23] For any $(n, M, n\delta)$-LMRM code it holds that

$$R \leqslant 2 - 2\delta \left\lfloor \frac{1}{\delta} \right\rfloor - \left(\delta \left\lfloor \frac{1}{\delta} \right\rfloor - \delta\right) \log_2(\delta) - \left(1 + \delta - \delta \left\lfloor \frac{1}{\delta} \right\rfloor\right) \log_2 \left(1 + \delta - \delta \left\lfloor \frac{1}{\delta} \right\rfloor\right) + o(1).$$

Lemma 23 [39, Thm. 27] For any $0 < \delta \leqslant 1$ the construction of [39, Cst. 1, Thm. 2] and [27, Sec. III-A] yields codes with

$$R = \left(1 - \delta \left\lfloor \frac{1}{\delta} \right\rfloor\right) \log_2 \left(\left\lceil \frac{1}{\delta} \right\rceil !\right) + \left(\delta + \delta \left\lfloor \frac{1}{\delta} \right\rfloor - 1\right) \log_2 \left(\left\lfloor \frac{1}{\delta} \right\rfloor !\right).$$

Previous works have also established the following lower bound on achievable rates of error-correcting codes:

Lemma 24 (Gilbert-Varshamov) For any $0 < \delta \leqslant 1$ there exist (n, M, d)-LMRM codes satisfying $\frac{d}{n} \geqslant \delta$ with rate $R \geqslant f_{\mathrm{GV}}(\delta) - o(1)$, where $\Phi(\delta) \leqslant f_{\mathrm{GV}}(\delta) \leqslant \varphi(\delta)$,

$$\Phi(\delta) = \begin{cases} \log_2 \frac{1}{\delta} + 2\delta \left(\log_2(e) - 1\right) - 1 & 0 < \delta \leqslant \frac{1}{2} \\ -2\delta \log_2 \frac{1}{\delta} + 2(1-\delta)\log_2(e) & \frac{1}{2} < \delta \leqslant 1, \end{cases}$$

$$\varphi(\delta) = \begin{cases} \log_2 \frac{1}{\delta} + \delta - 1 & 0 < \delta \leqslant \rho \\ \log_2 \frac{1}{\delta} + 2\delta \left(\log_2(e) - 1\right) - \log_2\left(e \log_2(e)\right) + 1 & \rho < \delta \leqslant \frac{1}{2} \\ \log_2 \left(\frac{\hat{t}(\delta)}{\log_2(e)}\right) - \hat{t}(\delta) \cdot (2\delta - 1) + \log_2 \frac{1}{1-\delta} & \frac{1}{2} < \delta \leqslant 1, \end{cases}$$

$\rho = \frac{2 - \log_2(e \log_2(e))}{3 - 2\log_2(e)}$, $W(t)$ is the Lambert function, and

$$\hat{t}(\delta) = \log_2(e) \cdot \left(\frac{2(1-\delta)}{2\delta - 1} - W\left(\frac{(1-\delta) \cdot \exp\left(\frac{2(1-\delta)}{2\delta - 1}\right)}{2\delta - 1}\right)\right).$$

Proof: We derive f_{GV} from the Gilbert-Varshamov bound:

$$f_{\mathrm{GV}}(\delta) = \frac{1}{n} \log_2 \left(\frac{n!}{\left|B_{\delta n,n}\right|}\right) = \log_2(n) - \log_2(e) - \frac{1}{n} \log_2(\left|B_{\delta n,n}\right|) + o(1),$$

where $\left|B_{\delta n,n}\right|$ is the size of ball of radius δn in S_n, and is independent of the center of the ball since the ℓ_∞ metric is right-invariant, i.e., for all $\sigma, \pi, \tau \in S_n$,

$$d_\infty(\sigma\pi, \tau\pi) = d_\infty(\sigma, \tau).$$

Unfortunately, the asymptotic size of $B_{\delta n,n}$ isn't precisely known as $n \to \infty$. Recently, however, [35] proved new lower bounds on $\left|B_{\delta n,n}\right|$, namely:

$$n \log_2(n) - \log_2(\left|B_{\delta n,n}\right|) \leqslant \begin{cases} n\left[\log_2 \frac{e}{\delta} + \delta - 1\right] - o(n) & 0 < \delta \leqslant \rho \\ n\left[\log_2 \frac{e}{\delta} + 2\delta\left(\log_2(e) - 1\right) - \log_2\left(e\log_2(e)\right) + 1\right] - o(n) & \rho < \delta \leqslant \frac{1}{2} \\ n\left[\log_2\left(\frac{e \cdot \hat{t}(\delta)}{\log_2(e)}\right) - \hat{t}(\delta) \cdot (2\delta - 1) + \log_2 \frac{1}{1-\delta}\right] - o(n) & \frac{1}{2} < \delta \leqslant 1. \end{cases}$$

An upper bound for $\left|B_{\delta n,n}\right|$ was established in [26, Eq. (4)] and [39, Lem. 25]:

$$n \log_2(n) - \log_2(\left|B_{\delta n,n}\right|) \geqslant \begin{cases} n\left[\log_2 \frac{1}{\delta} + (2\delta + 1)\left(\log_2(e) - 1\right)\right] - o(n) & 0 < \delta \leqslant \frac{1}{2} \\ n\left[(3 - 2\delta)\log_2(e) - 2\delta \log_2 \frac{1}{\delta}\right] - o(n) & \frac{1}{2} < \delta \leqslant 1. \end{cases}$$

Deriving the lemma is now straightforward. ■

The works cited in Lemma 24 establish a narrow rate-range for the Gilbert-Varshamov bound, as can be seen in Figure 3, i.e., the true Gilbert-Varshamov bound passes somewhere within the gray-shaded area in Figure 3.

Next, we aim to show that our construction can bridge some of the gap between the given bounds and known constructions.

Lemma 25 Let C_1 be the code from (the general version of) Construction A. Then an estimate from below of its rate R as a function of its normalized distance δ is given by (3).

Proof: The proof follows by a simple substitution of $(n \bmod d) = n - d\left\lfloor \frac{n}{d} \right\rfloor$ and $d = n\delta$ into (2). We also recall that $\rho_k > \frac{k-3}{k}$. ■

In conclusion, these asymptotic rates and bounds are shown in Figure 3. We note in particular that the rate of codes produced by Construction A is strictly higher than that of previously known constructions (as in Lemma 23). Furthermore, it produces codes with rates higher than those guaranteed by the Gilbert-Varshamov bound shown in Lemma 24 for all δ greater than ≈ 0.1 except in a small neighborhood of $\frac{1}{5}$, whereas known constructions only bypassed these rates for δ greater than ≈ 0.349.

V. Decoding Algorithm

This section is devoted to devising a decoding algorithm capable of correcting a noisy received version of a transmitted codeword.

Known constructions of (n, M, d)-LMRM codes, presented in [39, Cst. 1, Thm. 2] and [27, Sec. III-A], lend themselves to straightforward decoding algorithms, efficiently done in $O(n)$ operations, since for any given codeword σ and index $i \in [n]$, $r = [\sigma(i) \bmod d]$ is known. Hence, if a retrieved permutation τ satisfies $d_\infty(\sigma, \tau) \leqslant \lfloor (d-1)/2 \rfloor$, then $\sigma(i)$ is known to be the unique element $k \in r + d\mathbb{Z}$ satisfying $|k - \tau(i)| \leqslant \lfloor (d-1)/2 \rfloor$.

Our proposed construction diverges from that rigid partition. However, we can still efficiently decode noisy information, provided errors of magnitude no more than t have occurred, where $2t + 1 \leqslant d$. More precisely, we assume that for every stored permutation σ and retrieved permutation τ it holds that $d_\infty(\sigma, \tau) \leqslant t \leqslant \lfloor (d-1)/2 \rfloor$.

To simplify our presentation we assume $n = kd$, since then our construction only makes (repeated) use of a single auxiliary $G_\uparrow^{\mathrm{aux}}(k+1, \tilde{M}_{k+1})$ code. Extensions to the general version of Construction A are easily obtainable.

We first require a function `ValidAux` capable of detecting whether a given permutation $\pi \in S_{k+1}$ belongs to the auxiliary $G_\uparrow^{\mathrm{aux}}(k+1, \tilde{M}_{k+1})$ code.

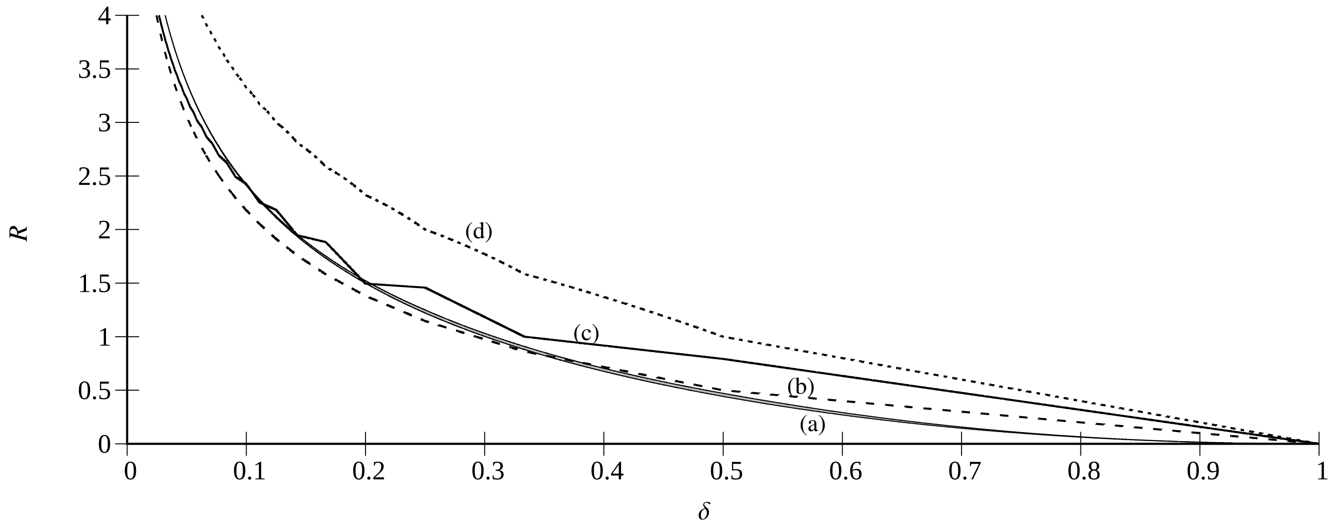

Figure 3. (a) The range of uncertainty for the Gilbert-Varshamov lower bound seen in Lemma 24. (b) The rate of codes from Lemma 23 constructed in [39]. (c) A lower bound for the rate of codes C_1 from Construction A. (d) The upper bound of Lemma 22.

$$R \geqslant \begin{cases} \left(1-\delta\left\lfloor\frac{1}{\delta}\right\rfloor\right)\log_2\left(\left\lceil\frac{1}{\delta}\right\rceil+1\right)+\delta\log_2\left(\left(\left\lfloor\frac{1}{\delta}\right\rfloor+1\right)!\right) & \\ \qquad +\left(1-\delta\left\lfloor\frac{1}{\delta}\right\rfloor\right)\log_2\left(\frac{\lceil 1/\delta\rceil-2}{\lceil 1/\delta\rceil+1}\right)-\delta & 4<\lfloor 1/\delta\rfloor\equiv 0 \pmod 2, \\ (1-4\delta)\left(\log_2(178)\right)+(5\delta-1)\log_2(57) & \lfloor 1/\delta\rfloor = 4, \\ (1-2\delta)\left(3-\log_2(3)\right)+\delta\log_2(3) & \lfloor 1/\delta\rfloor = 2, \\ \left(1-\delta\left\lfloor\frac{1}{\delta}\right\rfloor\right)\log_2\left(\left\lceil\frac{1}{\delta}\right\rceil+1\right)+\delta\log_2\left(\left(\left\lfloor\frac{1}{\delta}\right\rfloor+1\right)!\right) & \\ \qquad +\left(\delta+\delta\left\lfloor\frac{1}{\delta}\right\rfloor-1\right)\log_2\left(\frac{\lfloor 1/\delta\rfloor-2}{\lfloor 1/\delta\rfloor+1}\right)-\delta & 5<\lfloor 1/\delta\rfloor\equiv 1 \pmod 2, \\ (1-5\delta)\log_2(315)+(6\delta-1)\log_2(89)+2-9\delta & \lfloor 1/\delta\rfloor = 5, \\ (1-3\delta)\left(\log_2(57)-4\right)+1 & \lfloor 1/\delta\rfloor = 3, \\ (1-\delta)\log_2(3) & \lfloor 1/\delta\rfloor = 1. \end{cases} \tag{3}$$

Lemma 26 For an auxiliary $G^{\mathrm{aux}}_{\uparrow}(k+1,\tilde{M}_{k+1})$ code provided by Lemma 7, Theorem 9 or Theorem 14, a function `ValidAux` can be implemented to operate in $O(k)$ steps.

Proof: If we use Lemma 7, then the auxiliary code consists of all even permutations, and it is well known that we can determine the signature of a permutation $\pi \in S_{k+1}$ in $O(k)$ operations, e.g., by finding a cycle decomposition of π. The case of $k+1=5$ requires special attention, as $\tilde{M}_5 = 57$. In fact, in that case [44] showed that a parity-preserving code of size 57 exists consisting of

$$A_5 \setminus \left\{\sigma, t_{\uparrow 3}\sigma, {t_{\uparrow 3}}^2\sigma\right\},$$

for every choice of $\sigma \in A_5$. The user may arbitrarily decide on σ, and check for the missing codewords in $O(k)$.

If we instead use Theorem 9, then we know that the vector notation of every codeword in the auxiliary code has 1 following $k+1$ (cyclically). Since there are

$$\frac{|S_{k+1}|}{k} = k! + (k-1)! = k!\left(1\cdot\frac{k-1}{k}+2\cdot\frac{1}{k}\right)$$

such codewords, we observe that the auxiliary code consists of precisely all permutations so characterized (when counting valid permutations, we partition permutations on $[k]$ by whether the first index in their vector notation equals 1. If so, we may insert $k+1$ either at the beginning or the end of their vector notation; otherwise, its position is uniquely determined), i.e.,

$$\pi\left(\left(\pi^{-1}(k+1) \bmod (k+1)\right)+1\right) = 1,$$

which again requires $O(k)$ steps to verify.

Finally, for Theorem 14 we note that the problem can equivalently be solved for the codes of Lemma 13, as composition with $\tilde{\pi}$ can be done in $O(1)$ steps with a simply implemented rule, or naively in at most $O(k)$ steps. If we divide into cases according to $\pi(k+1)=j$ we may identify r such that π must belong to P_r, or not belong to our code. For values $j=1,k$, we know π can only be a bridge; For $j=k+1$, it must belong to P'_0. In these cases, only cyclic shifts (on a subset of indices, by case) of a known permutation are valid, which we can easily verify in linear time. For all other elements P_r consists of all even permutations satisfying $\pi(k+1)=j$, hence the problem again reduces to determining $\operatorname{sign}\pi$, as in the case based on Lemma 7 (or managed as discussed above for $k=5$). ■

An important notion of a *window* will be useful. Let $\sigma \in S_n$ be a permutation, $n = kd$. For all $j \in [d]$ we define the jth window as the set of indices

$$W_j = \{k(j-1)+2, k(j-1)+3, \ldots, kj+1\} \cap [n].$$

The windows partition $[n] \setminus \{1\}$, and are all of size k except W_d which is of size $k-1$.

Given a set $I \subseteq [n]$, we conveniently denote

$$\sigma(I) = \{\sigma(i) \mid i \in I\}.$$

We prove a simple lemma concerning properties of windows of codewords from Construction A.

Lemma 27 Let σ be a codeword of C_1 from Construction A, with $n = kd$. Then for all $j \in [d]$,

$$k-1 \leqslant \left|\sigma(W_j) \cap \mathcal{R}_j\right| \leqslant k,$$

i.e., at most one element of $\sigma(W_j)$ does not leave a residue of j modulo d. In particular, $\sigma(W_d) \subseteq \mathcal{R}_d$.

Additionally, if $\left|\sigma(W_j) \cap \mathcal{R}_j\right| = k-1$, $j \in [d-1]$, and we denote $\{x\} = \sigma(W_j) \setminus \mathcal{R}_j$, then there exists some $j' > j$ such that $x \in \mathcal{R}_{j'}$.

Proof: Take any $1 < j \in [d]$, and let σ_j be the C_j-parent of σ. Then, in C_1, no transition between σ and σ_j is induced by C_j, and hence σ_j is derived from σ by a (perhaps empty) sequence of $t_{\uparrow i'}$ transitions, for $i' \in W_1 \cup \cdots \cup W_{j-1}$. Therefore, for all $i \in W_j \cup \cdots \cup W_d$ we have $\sigma_j(i) = \sigma(i)$, and the same also holds for $j = 1$ (since $\sigma_1 = \sigma$). In particular, $\sigma(W_j) = \sigma_j(W_j)$.

Now, since C_j only applies "push-to-the-$k(j-1)+1$st-index" transitions, and

$$\sigma_0\left(\{k(j-1)+1\} \cup W_j \cup \cdots \cup W_d\right) = \mathcal{R}_j \cup \cdots \cup \mathcal{R}_d,$$

if for any $i \in W_j$ we have $\sigma(i) = \sigma_j(i) \notin \mathcal{R}_j$, then by necessity $\sigma(i) \in \mathcal{R}_{j'}$ for some $j' > j$. In particular, $\sigma(W_d) \subseteq \mathcal{R}_d$.

For all $j \in [d-1]$, we also consider σ_{j+1}, the C_{j+1}-parent of both σ and σ_j. Since C_{j+1} only applies "push-to-the-$kj+1$st-index" transitions,

$$\begin{aligned}
&\sigma_{j+1}\left((\{k(j-1)+1\} \cup W_j) \setminus \{kj+1\}\right) \\
&\quad = \sigma_0\left((\{k(j-1)+1\} \cup W_j) \setminus \{kj+1\}\right) = \mathcal{R}_j.
\end{aligned}$$

Finally, since σ_{j+1} is derived from σ_j by a sequence of $t_{k(j-1)+1\uparrow i'}$ transitions for $i' \in W_j$, it follows that

$$\sigma_{j+1}\left(\{k(j-1)+1\} \cup W_j\right) = \sigma_j\left(\{k(j-1)+1\} \cup W_j\right)$$

thus

$$\begin{aligned}
\sigma_j(W_j) &\subseteq \sigma_{j+1}\left(\{k(j-1)+1\} \cup W_j\right) \\
&= \mathcal{R}_j \cup \left\{\sigma_{j+1}(kj+1)\right\}.
\end{aligned}$$

Noting that $\left|\sigma_j(W_j)\right| = \left|\mathcal{R}_j\right| = k$ and recalling that $\sigma(W_j) = \sigma_j(W_j)$, we are done. ∎

Corollary 28 Let σ be a codeword of C_1 from Construction A, with $n = kd$. Then for each $j \in [d]$, there is a unique element $x_j^\sigma \in \mathcal{R}_j \cup \cdots \cup \mathcal{R}_d$ satisfying

$$\sigma(W_j \cup \cdots \cup W_d) = \mathcal{R}_j \cup \cdots \cup \mathcal{R}_d \setminus \left\{x_j^\sigma\right\}.$$

Proof: The proposition follows from Lemma 27 for $j = d$ since $|\sigma(W_d)| = |W_d| = k-1 \leqslant |\mathcal{R}_d \cap \sigma(W_d)|$. Now suppose the proposition holds for $j+1$, and we prove that it holds for j.

We again observe by Lemma 27 that $|\mathcal{R}_j \cap \sigma(W_j)| \in \{k-1, k\}$. If $|\mathcal{R}_j \cap \sigma(W_j)| = k$, since $|\sigma(W_j)| = |W_j| = k$, then $\mathcal{R}_j = \sigma(W_j)$ and $x_j^\sigma = x_{j+1}^\sigma$ satisfies the claim.

Otherwise $\sigma(W_j) \setminus \mathcal{R}_j = \{y\}$ for some $y \in [n]$; it would suffice to show $y = x_{j+1}^\sigma$, since then $\mathcal{R}_j \setminus \sigma(W_j) = \{x_j^\sigma\}$ would satisfy the claim.

Consider then σ_j, the C_j-parent of σ. Note that $\sigma_j(W_j) = \sigma(W_j)$, and since C_j employs "push-to-the-$(k(j-1)+1)$st-index" transitions only, and

$$\sigma_0(W_j \cup \cdots \cup W_d) \subseteq \mathcal{R}_j \cup \cdots \cup \mathcal{R}_d,$$

we know that $\sigma(W_j) \subseteq \mathcal{R}_j \cup \cdots \cup \mathcal{R}_d$. We now use the induction hypothesis

$$\sigma(W_{j+1} \cup \cdots \cup W_d) = \left(\mathcal{R}_{j+1} \cup \cdots \cup \mathcal{R}_d\right) \setminus \left\{x_{j+1}^\sigma\right\},$$

and it follows that $\sigma(W_j) \subseteq \mathcal{R}_j \cup \left\{x_{j+1}^\sigma\right\}$, hence $y = x_{j+1}^\sigma$. ∎

From now on, we denote $i_j^\sigma = \sigma^{-1}(x_j^\sigma)$. Another useful notation we shall employ is a function that quantizes any integer to the nearest integer leaving a residue of j modulo d. We denote this function by $q_d^j : \mathbb{Z} \to d\mathbb{Z} + j$, defined by

$$q_d^j(a) = \operatorname*{argmin}_{b \in d\mathbb{Z}+j} |a-b|,$$

where we assume argmin returns a single value, and ties are broken arbitrarily.

For the decoding procedure description, let us fix the parameters $n = kd$, and the code C_1 from Construction A. Additionally, we denote by $\sigma \in C_1$ the transmitted permutation, by $\tau \in S_n$ the received permutation, and by $\hat{\sigma} \in S_n$ the decoded permutation. We denote the *decoding radius* by $t = \lfloor (d-1)/2 \rfloor$, and assume $d_\infty(\sigma, \tau) \leqslant t$.

We will decode τ iteratively by window, from W_1 to W_d. We shall make sure–inductively–that when we begin the process of decoding W_j, for some $j \in [d]$, we know i_j^σ. Initially, as mentioned, we set $j = 1$. Trivially, $i_1^\sigma = 1$.

Step I We set the decoding window

$$\hat{W}_j = W_j \cup \left\{i_j^\sigma\right\},$$

and naively decode $\hat{W}_j$ by setting for all $i \in \hat{W}_j$,

$$\hat{\sigma}(i) = q_d^j(\tau(i)).$$

Lemma 29 After Step I, for all $i \in \hat{W}_j$ such that $\sigma(i) \in \mathcal{R}_j$ it holds that $\hat{\sigma}(i) = \sigma(i)$.

Proof: For all such i we have $\hat{\sigma}(i) \equiv \sigma(i) \pmod d$ and

$$\begin{aligned}
|\hat{\sigma}(i) - \sigma(i)| &\leqslant |\hat{\sigma}(i) - \tau(i)| + |\tau(i) - \sigma(i)| \\
&= \left|q_d^j(\tau(i)) - \tau(i)\right| + |\tau(i) - \sigma(i)| \\
&\leqslant \lfloor d/2 \rfloor + t < d.
\end{aligned}$$

∎

Corollary 30 After Step I,

$$\hat{\sigma}(\hat{W}_j) = \hat{\sigma}\left(\hat{W}_j \setminus \left\{i^{\sigma}_{j+1}\right\}\right) = \mathcal{R}_j.$$

Proof: By Corollary 28 we know that $\mathcal{R}_j \subseteq \sigma\left(\hat{W}_j\right)$. We further recall that $\sigma(i^{\sigma}_{j+1}) \notin \mathcal{R}_j$, hence

$$\mathcal{R}_j \subseteq \sigma\left(\hat{W}_j \setminus \left\{i^{\sigma}_{j+1}\right\}\right),$$

and since

$$\left|\sigma\left(\hat{W}_j \setminus \left\{i^{\sigma}_{j+1}\right\}\right)\right| = \left|\hat{W}_j \setminus \left\{i^{\sigma}_{j+1}\right\}\right| = k = \left|\mathcal{R}_j\right|$$

we have equality. The claim now follows from Lemma 29. ∎

Corollary 30 implies that after Step I, $\hat{\sigma}(\hat{W}_j)$ contains a unique element of $\mathcal{R}_j$ which appears twice, and every other element appears exactly once; by Lemma 29 these other elements have been decoded correctly. Before we can continue inductively to decode W_{j+1}, it only remains to find i^{σ}_{j+1}; the other instance in $\hat{W}_j$ of $\hat{\sigma}(i^{\sigma}_{j+1})$ we therefore also know to have been decoded correctly.

We shall identify i^{σ}_{j+1} using C^{aux}, the auxiliary $G^{\text{aux}}_{\uparrow}(k+1, \tilde{M}_{k+1})$ code used in Construction A. By construction, if we examine σ_j, the C_j-parent of σ, then for all $i \in W_j$ we observe $\sigma(i) = \sigma_j(i)$, and $\sigma(i^{\sigma}_j) = \sigma_j(k(j-1)+1)$. The ordering of the $k+1$ elements of $\sigma(\hat{W}_j) = \sigma_j\left(\{k(j-1)+1\} \cup W_j\right)$ is then induced by a permutation of C^{aux}. We construct this induced permutation from the auxiliary code C^{aux}, which we denote $\hat{\pi} \in S_{k+1}$. We first define a simple bijection $\alpha_j : \mathcal{R}_j \to [k]$, which is the inverse of the enumeration of $\mathcal{R}_j$ given by the arbitrary initial order of elements in σ_0 used in Construction A, e.g., in the simple case $n = kd$,

$$\alpha_j(m) = \begin{cases} \left\lfloor \frac{m}{d} \right\rfloor & j < m \in \mathcal{R}_j, \\ k & m = j. \end{cases}$$

With α_j we define $\hat{\pi}$ as,

$$\hat{\pi}(i) = \begin{cases} \alpha_j(\hat{\sigma}(i^{\sigma}_j)) & i = 1; \\ \alpha_j(\hat{\sigma}(k(j-1)+i)) & i \in \{2, 3, \ldots, k+1\}, \end{cases}$$

and note that–as it currently stands–$\hat{\pi}$ is not a permutation of $[k+1]$ because its range is $[k]$ and some unique $a \in [k]$ has two distinct pre-images.

Theorem 31 Let $s, t \in [k+1]$ be the unique pair of indices such that $\hat{\pi}(s) = \hat{\pi}(t) = a \in [k]$. There is a unique way to re-define $\hat{\pi} \restriction_{\{s,t\}}$ (the restriction of $\hat{\pi}$ to $\{s,t\}$) as a bijection onto $\{a, k+1\}$ that yields $\hat{\pi} \in C^{\text{aux}}$. Furthermore, if we define $I_j : [k+1] \times [n] \to [n]$ by

$$I_j(q, r) = \begin{cases} r & q = 1, \\ k(j-1) + q & \text{otherwise} \end{cases}$$

then after performing that correction

$$i^{\sigma}_{j+1} = I_j(\hat{\pi}^{-1}(k+1), i^{\sigma}_j).$$

Proof: First, arbitrarily set $\hat{\pi}(t) = k+1$, where $t > s$. Once corrected, $\hat{\pi} \in S_{k+1}$ by Corollary 30 and because $\alpha_j : \mathcal{R}_j \to [k]$ is a bijection.

Now, we take $\pi \in C^{\text{aux}}$ which generates σ_j in the recursion step of Construction A–while constructing C_j–from its C_{j+1}-parent. Hence

$$\pi(i) = \begin{cases} \alpha_j(x^{\sigma}_j) & i = 1, \\ \alpha_j(\sigma(k(j-1)+i)) & i \in \{2, 3, \ldots, k+1\}, \end{cases}$$

and therefore either $\hat{\pi} = \pi$ or $\hat{\pi} = (k+1, a) \circ \pi$. Crucially, we observe that in the latter case $\hat{\pi} \notin C^{\text{aux}}$ since C^{aux} is a $G^{\text{aux}}_{\uparrow}(k+1, \tilde{M}_{k+1})$ code and $\pi \in C^{\text{aux}}$; we utilize `ValidAux` to discover whether our original arbitrary correction should be reversed.

To complete the proof, we note by the recursion step of Construction A that, indeed, $i^{\sigma}_{j+1} = I_j(\pi^{-1}(k+1), i^{\sigma}_j)$. ∎

We can therefore complete our iterative decoding round with the following step.

Step II We construct $\hat{\pi}$ as described, identify s, t, $s < t$, and arbitrarily correct $\hat{\pi}(t) = k+1$. We test `ValidAux`($\hat{\pi}$): if true, we have $i^{\sigma}_{j+1} = I_j(t, i^{\sigma}_j)$; otherwise, it holds that $i^{\sigma}_{j+1} = I_j(s, i^{\sigma}_j)$.

Finally, observe that when decoding W_d it's known that $\sigma(\hat{W}_d) = \mathcal{R}_d$, hence by Lemma 29 $\hat{W}_d$ is decoded correctly, and we need not (and–indeed–cannot) perform Step II.

Example 32 We shall demonstrate the decoding process assuming once again $n = kd$ for simplicity, and using the parameters $d = 3$ (hence $t = 1$), $k = 2$ and code constructed in Example 19. Recall that the $G^{\text{aux}}_{\uparrow}(3,3)$ code used in that example is

$$C^{\text{aux}} = \{[1,2,3],\ [3,1,2],\ [2,3,1]\}.$$

We choose the transmitted codeword $\sigma = [1,2,4,6,5,3]$, and a noisy received permutation $\tau = [1,3,4,5,6,2]$.

We start by defining $i_1 = 1$ and observing (by abuse of the vector notation) $\tau \restriction_{\hat{W}_1} = [1;3,4]$ (the first element is differentiated because–generally although never when $j = 1$–it does not immediately precede the rest in τ's vector notation).

Since $j = 1$, we define $\hat{\sigma} \restriction_{\hat{W}_1} = [1;4,4]$. This leads us to construct $\hat{\pi} = [2,1,3] \notin C^{\text{aux}}$, so we instead correct $\hat{\pi} = [2,3,1]$ and define $i_2 = 2$. (So far we have $\hat{\sigma} = \left[1, \underline{4}, 4, \cdot, \cdot, \cdot\right]$, where an underline marks i^{σ}_{j+1}.)

Next, we have $\tau \restriction_{\hat{W}_2} = [3;5,6]$, which ($j = 2$) we decode $\hat{\sigma} \restriction_{\hat{W}_1} = [2;5,5]$. This again generates $\hat{\pi} = [2,1,3] \notin C^{\text{aux}}$, and we correct in similar fashion to $\hat{\pi} = [2,3,1]$ and define $i_2 = 4$. (Up to this point, we have $\hat{\sigma} = \left[1, \cancel{4}2, 4, \underline{5}, 5, \cdot\right]$).

Finally, we have $\tau \restriction_{\hat{W}_3} = [5;2]$ and since $j = 3$ we decode $\hat{\sigma} \restriction_{\hat{W}_1} = [6;3]$, and overall $\hat{\sigma} = \left[1,2,4,\cancel{5}6,5,3\right] = \sigma$. □

Example 33 We present another example, intended to demonstrate the process in more detail, for which we depart from the parameters used in Example 19 by setting $d = 5$ (allowing for $t = 2 \leqslant \lfloor (d-1)/2 \rfloor$), $k = 3$. In each recursion step of Construction A the $G^{\text{aux}}_{\uparrow}(4,8)$ code used is C^{aux} presented in Example 10.

The codeword

$$\sigma = [11,1,8,6,7,2,12,13,3,5,9,14,4,10,15]$$

appears in the code generated in this case, as can be seen by identifying its C_5, C_4, C_3, and C_2 parents as, respectively,

$$\begin{aligned}
\sigma_5 &= [6, 11, 1, 7, 12, 2, 8, 13, 3, 9, 14, 4, 5, 10, 15],\\
\sigma_4 &= [6, 11, 1, 7, 12, 2, 8, 13, 3, 5, 9, 14, 4, 10, 15],\\
\sigma_3 &= \sigma_4,\\
\sigma_2 &= [6, 11, 1, 8, 7, 2, 12, 13, 3, 5, 9, 14, 4, 10, 15].
\end{aligned}$$

We choose

$$\tau = [12, 3, 9, 7, 5, 2, 11, 15, 1, 6, 8, 13, 4, 10, 14]$$

to be the noisy version of the transmitted codeword σ, and verify that $d_\infty(\tau, \sigma) = 2 = t$.

Beginning with $j = 1$, we have $\tau \restriction_{\hat{W}_1} = [12; 3, 9, 7]$, which we decode $\hat{\sigma} \restriction_{\hat{W}_1} = [11; 1, 11, 6]$, generating $\hat{\pi} = [2, 3, 2, 1]$ which is corrected to $\hat{\pi} = [2, 3, 4, 1] \in C^{\text{aux}}$. We identify $i_2 = 3$, and keep

$$\hat{\sigma} = \left[11, 1, \underline{11}, 6, \cdot, \cdot, \cdot, \cdot, \cdot, \cdot, \cdot, \cdot, \cdot, \cdot, \cdot\right].$$

Next, for $j = 2$, observe that $\tau \restriction_{\hat{W}_2} = [9; 5, 2, 11]$, and we decode $\hat{\sigma} \restriction_{\hat{W}_2} = [7; 7, 2, 12]$. This generates $\hat{\pi} = [1, 1, 3, 2]$, which we initially correct to $\hat{\pi} = [1, 4, 3, 2] \notin C^{\text{aux}}$, so (skip correcting $\hat{\pi}$, as it has no further consequence) $i_3 = i_2 = 3$ instead of $i_3 = 4$. We summarize

$$\hat{\sigma} = \left[11, 1, \cancel{11}\underline{7}, 6, 7, 2, 12, \cdot, \cdot, \cdot, \cdot, \cdot, \cdot, \cdot, \cdot\right].$$

We turn to $\hat{W}_3$ and see that $\tau \restriction_{\hat{W}_3} = [9; 15, 1, 6]$, decoded to $\hat{\sigma} \restriction_{\hat{W}_3} = [8; 13, 3, 8]$. We generate $\hat{\pi} = [1, 2, 3, 1]$ and correct it to $\hat{\pi} = [1, 2, 3, 4] \in C^{\text{aux}}$, indicating that $i_4 = 10$. We now have

$$\hat{\sigma} = \left[11, 1, \cancel{7}8, 6, 7, 2, 12, 13, 3, \underline{8}, \cdot, \cdot, \cdot, \cdot, \cdot\right].$$

Moving on to $j = 4$, while decoding $\hat{W}_4$ we note $\tau \restriction_{\hat{W}_4} = [6; 8, 13, 4]$, which we decode as $\hat{\sigma} \restriction_{\hat{W}_4} = [4; 9, 14, 4]$. This generates $\hat{\pi} = [3, 1, 2, 3]$ which is corrected to $\hat{\pi} = [3, 1, 2, 4] \notin C^{\text{aux}}$. We therefore define $i_5 = i_4 = 10$ instead of $i_5 = 13$. Up to now,

$$\hat{\sigma} = \left[11, 1, 8, 6, 7, 2, 12, 13, 3, \cancel{8}\underline{4}, 9, 14, 4, \cdot, \cdot\right].$$

Finally, $j = 5$, and we get $\tau \restriction_{\hat{W}_5} = [4; 10, 14]$ which we decode to $\hat{\sigma} \restriction_{\hat{W}_5} = [5; 10, 15]$, and overall

$$\hat{\sigma} = \left[11, 1, 8, 6, 7, 2, 12, 13, 3, \cancel{4}5, 9, 14, 4, 10, 15\right] = \sigma.$$

□

The decoding algorithm is formalized in `Decode`(τ). With appropriate simple data structures, the algorithm requires $O(kd) = O(n)$ steps. We assume simple integer operations to take constant-time.

VI. Ranking and Unranking

In this section we discuss the process of encoding data $m \in \{0, 1, \ldots, |C_1| - 1\}$ to a codeword $\sigma \in C_1$, which is also known as *unranking* m, and the inverse process of *ranking* $\sigma \in C_1$, i.e., obtaining its rank in the code. Throughout this section, C_1 stands for the code obtained via Construction A.

Due to the nature of our construction, performing these tasks with the codes generated by Theorem 18 is reliant on our

Function `Decode`(τ)

```
input  : τ ∈ S_{kd} satisfying d_∞(τ, C_1) ⩽ t ⩽ ⌊(d − 1)/2⌋.
output : σ̂ ∈ C_1 such that d_∞(τ, σ̂) ⩽ t.
i ← 1
for j = 1, 2, ..., d − 1 do
    /* Naively decode Ŵ_j */
    σ̂(i) ← q_d^j(τ(i))
    π̂(1) ← α_j(σ̂(i))
    for r = 2, ..., k + 1 do
        m ← q_d^j(τ(k(j − 1) + r))
        if σ̂^{−1}(m) is already set then
            π̂(r) ← k + 1
            a ← α_j(m)
        else
            π̂(r) ← α_j(m)
        σ̂(r) ← m
    /* Define i_{j+1} */
    if ValidAux(π̂) then
        i ← I(π̂^{−1}(k + 1), i)
    else
        i ← I(π̂^{−1}(a), i)
/* Decode Ŵ_d */
σ̂(i) ← q_d^d(τ(i))
for r = 2, ..., k do
    σ̂(r) ← q_d^j(τ(k(d − 1) + r))
return σ̂
```

ability to do the same with the codes provided by Lemma 8 and Corollary 15. We therefore recall the following known result.

Lemma 34 [23] The complete $G_\uparrow(n, n!)$ codes provided by Lemma 8 has a ranking algorithm operating in $O(n)$ steps, and an unranking scheme operating in $O(n^2)$ steps.

This gives rise to the following corollary.

Corollary 35 The $G_\uparrow^{\text{aux}}(2m, \frac{|S_{2m}|}{2m-1})$ codes generated by Theorem 9 can be ranked in $O(m)$ operations and unranked in $O(m^2)$ operations.

Proof: Ranking a permutation σ in the code may proceed by finding the cyclic shift required for $[2m, 1]$ to be the first two elements. After removing these two first elements, and then reversing the permutation we may use a ranking algorithm from Lemma 34. A simple combination of the results gives the required ranking of σ. By Lemma 34, the entire procedure takes $O(m)$ operations. A symmetric argument gives an $O(m^2)$ algorithm for unranking. ■

Unfortunately, no ranking and unranking schemes are known for parity-preserving $G_\uparrow(2m + 1, M_{2m+1})$ codes provided by Lemma 7 (developed in [20]), or previous constructions presented in [22], [45]. Consequentially, we rely on Theorem 9 instead of Theorem 14 for even sized congruence classes. In the case of odd sized classes, we can leverage the following codes.

Lemma 36 [44] For all $m \geqslant 1$ there exist parity-preserving $G_\uparrow(2m + 1, \hat{M}_{2m+1})$ codes with sizes

$$\hat{M}_{2m+1} = \left(\frac{(2m)!}{m!}\right)^2 \frac{(2m + 1)}{2^{2m}} = \frac{(2m)!}{m!^2 2^{2m}} |S_{2m+1}|.$$

These codes can be ranked and unranked in $O(m^2)$ operations.

We summarize those observations in the following corollary.

Corollary 37 For all $k \geqslant 3$ there exist a $G_{\uparrow}^{\text{aux}}(k, \hat{M}_k)$ code, which have ranking and unranking schemes operating in $O(k^2)$ steps, where

$$\hat{M}_k = \begin{cases} \frac{(k-1)!}{((k-1)/2)!^2 2^{k-1}} k! & k \equiv 1 \pmod 2; \\ \frac{k!}{(k-1)} & k \equiv 0 \pmod 2. \end{cases}$$

Note that we can now replace Corollary 15 by Corollary 37 in Construction A to obtain codes which we shall denote $\hat{C}_1$, and each auxiliary code on a congruence class of size $k > 1$ contributes to $\left|\hat{C}_1\right|$ a multiplicative factor of

$$\hat{M}_{k+1} = \begin{cases} \frac{k!}{(k/2)!^2 2^k}(k+1)! & k \equiv 0 \pmod 2, \\ \frac{(k+1)!}{k} & k \equiv 1 \pmod 2. \end{cases}$$

We also note, using Stirling's approximation

$$e^{\frac{1}{12n+1}} < \frac{n! e^n}{n^n \sqrt{2\pi n}} < e^{\frac{1}{12n}},$$

that

$$\frac{k!}{(k/2)!^2 2^k} > \sqrt{\frac{2}{\pi k}} \left(e^{1-\frac{1}{4+\frac{1}{3k}}}\right)^{-\frac{1}{3k}} > \sqrt{\frac{2}{\pi k}} e^{-1/(4k)}.$$

We can then recalculate code size, in the case of $\left\lfloor \frac{n}{d} \right\rfloor \equiv 0 \pmod 2$:

$$\begin{aligned} \left|\hat{C}_1\right| &= \left(\frac{(\lceil n/d \rceil + 1)!}{\lceil n/d \rceil}\right)^{n \bmod d} \cdot \left\lfloor \frac{n}{d} \right\rfloor! \\ &\quad \cdot \left(\frac{\lfloor n/d \rfloor!}{(\lfloor n/d \rfloor /2)!^2 2^{\lfloor n/d \rfloor}} \left(\left\lfloor \frac{n}{d} \right\rfloor + 1\right)!\right)^{d-(n \bmod d)-1} \\ &> \left\lceil \frac{n}{d} \right\rceil!^{\,n \bmod d} \left\lfloor \frac{n}{d} \right\rfloor!^{\,d-(n \bmod d)} \\ &\quad \cdot \left(1 + \frac{1}{\lceil n/d \rceil}\right)^{n \bmod d} \cdot \left(\left\lfloor \frac{n}{d} \right\rfloor + 1\right)^{(d-1)-(n \bmod d)} \\ &\quad \cdot \left(\sqrt{\frac{2}{\pi \lfloor n/d \rfloor}} e^{-1/(4\lfloor n/d \rfloor)}\right)^{(d-1)-(n \bmod d)}, \end{aligned}$$

and when $\left\lfloor \frac{n}{d} \right\rfloor \equiv 1 \pmod 2$:

$$\begin{aligned} \left|\hat{C}_1\right| &= \left(\frac{\lceil n/d \rceil!}{(\lceil n/d \rceil /2)!^2 2^{\lceil n/d \rceil}} \left(\left\lceil \frac{n}{d} \right\rceil + 1\right)!\right)^{n \bmod d} \\ &\quad \cdot \left\lfloor \frac{n}{d} \right\rfloor! \cdot \left(\frac{(\lfloor n/d \rfloor + 1)!}{\lfloor n/d \rfloor}\right)^{d-(n \bmod d)-1} \\ &> \left\lceil \frac{n}{d} \right\rceil!^{\,n \bmod d} \left\lfloor \frac{n}{d} \right\rfloor!^{\,d-(n \bmod d)} \\ &\quad \cdot \left(\left\lceil \frac{n}{d} \right\rceil + 1\right)^{n \bmod d} \cdot \left(1 + \frac{1}{\lfloor n/d \rfloor}\right)^{(d-1)-(n \bmod d)} \\ &\quad \cdot \left(\sqrt{\frac{2}{\pi \lceil n/d \rceil}} e^{-1/(4\lceil n/d \rceil)}\right)^{n \bmod d}, \end{aligned}$$

and we note that in the special case $\left\lfloor \frac{n}{d} \right\rfloor = 1$ we have $\left|\hat{C}_1\right| = |C_1|$.

Function Rank(σ)

input : $\sigma \in \hat{C}_1$.
output : $m \in \left\{0, 1, \ldots, \left|\hat{C}_1\right| - 1\right\}$ which is the rank of σ in $\hat{C}_1$.
/* Build a permutation $\pi_d \in S_k$ */
for $i \in W_d$ **do**
  $\pi_d[i - k(d-1)] \leftarrow \alpha_d(\sigma[i])$
$\pi_d[1] \leftarrow [k] \setminus \pi_d[2, \ldots, k]$
$m \leftarrow ((\texttt{RankComplete}(\pi_d) - 1) \bmod k!)$
for $j = d-1, \ldots, 1$ **do**
  /* Build a permutation $\pi_j \in S_{k+1}$ */
  for $i \in W_j$ **do**
    $\pi_j[i - k(j-1)] \leftarrow \alpha_j(\sigma[i])$
  $\pi_j[1] \leftarrow [k+1] \setminus \pi_j[2, \ldots, k+1]$
  $m \leftarrow m \cdot \hat{M}_{k+1} + \left((\texttt{RankAux}(\pi_j) - 1) \bmod \hat{M}_{k+1}\right)$
return $(m+1) \bmod \left|\hat{C}_1\right|$

We likewise observe the rates of codes based on Corollary 37, and find for $\lfloor 1/\delta \rfloor \equiv 0 \pmod 2$

$$\begin{aligned} \hat{R} \geqslant &\left(1 - \delta \left\lfloor \frac{1}{\delta} \right\rfloor\right) \log_2 \left(\left\lceil \frac{1}{\delta} \right\rceil! \left(1 + \frac{1}{\lceil 1/\delta \rceil}\right)\right) \\ &+ \left(\delta + \delta \left\lfloor \frac{1}{\delta} \right\rfloor - 1\right) \log_2 \left(\left(\left\lfloor \frac{1}{\delta} \right\rfloor + 1\right)!\right) \\ &- \frac{1}{2}\left(\delta + \delta \left\lfloor \frac{1}{\delta} \right\rfloor - 1\right) \\ &\cdot \left(\log_2 \left\lfloor \frac{1}{\delta} \right\rfloor + \frac{\log_2(e)}{2 \lfloor 1/\delta \rfloor} + \log_2(\pi) - 1\right) - o(1), \end{aligned}$$

and for $\lfloor 1/\delta \rfloor \equiv 1 \pmod 2$

$$\begin{aligned} \hat{R} \geqslant &\left(1 - \delta \left\lfloor \frac{1}{\delta} \right\rfloor\right) \log_2 \left(\left(\left\lceil \frac{1}{\delta} \right\rceil + 1\right)!\right) \\ &+ \left(\delta + \delta \left\lfloor \frac{1}{\delta} \right\rfloor - 1\right) \log_2 \left(\left\lfloor \frac{1}{\delta} \right\rfloor! \left(1 + \frac{1}{\lfloor 1/\delta \rfloor}\right)\right) \\ &- \frac{1}{2}\left(1 - \delta \left\lfloor \frac{1}{\delta} \right\rfloor\right) \\ &\cdot \left(\log_2 \left\lceil \frac{1}{\delta} \right\rceil + \frac{\log_2(e)}{2 \lceil 1/\delta \rceil} + \log_2(\pi) - 1\right) - o(1). \end{aligned}$$

The losses in asymptotic rate are shown in Figure 4. We observe in particular that we still manage to achieve better rates than previously known error-correcting codes (without the Gray property), even with the significantly smaller $G_{\uparrow}^{\text{aux}}(k, \hat{M}_k)$ of Corollary 37.

Let us denote by RankComplete(π), UnrankComplete(m) the ranking and unranking procedures for the complete codes from Lemma 34. Additionally, let RankAux(π) and UnrankAux(m) denote the ranking and unranking procedures for the auxiliary codes of Corollary 37. We can readily take advantage of $\hat{C}_1$'s tiered structure to use these functions in order to perform the same tasks for our construction. We include pseudo-code for these algorithms, which we call Rank(σ) and Unrank(m), for completeness. As before, we assume $n = kd$ to simplify the presentation.

Theorem 38 For the code $\hat{C}_1$ of length $n = kd$, the algorithms Rank(σ), Unrank(m) operate in $O(k^2 d)$ steps.

Proof: Both algorithms perform a single loop over all indices of σ, making simple integer operations, which requires

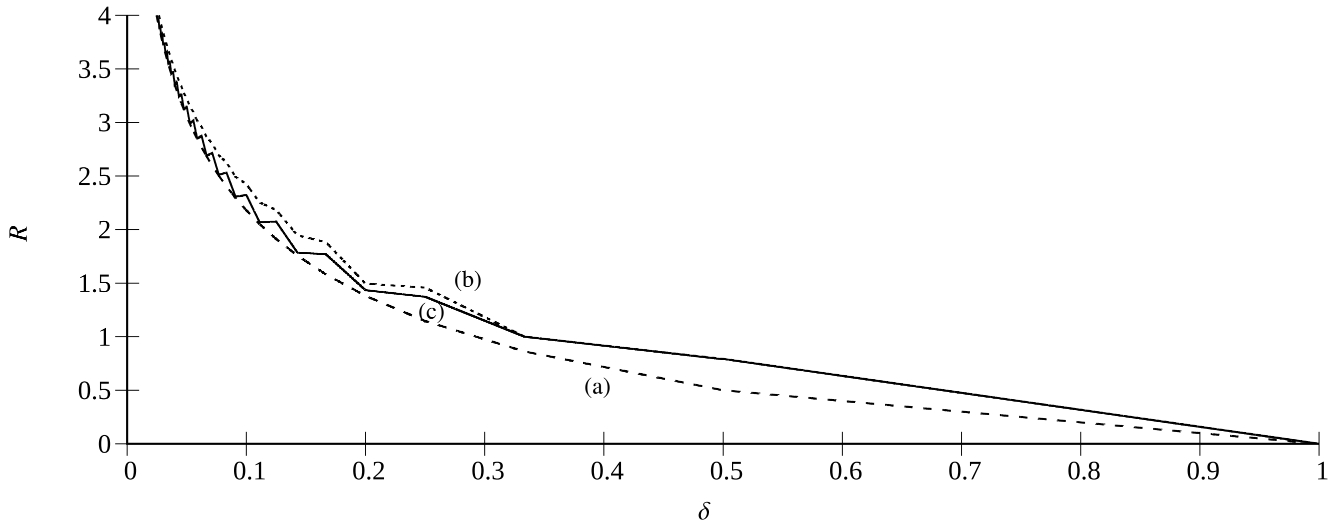

Figure 4. (a) The rate of codes from Lemma 23 constructed in [39]. (b) The rate of codes C_1 from Construction A. (c) The rate of codes $\hat{C}_1$ constructed using auxiliary codes from Corollary 37.

Function `Unrank` (m)

```
input  : m ∈ {0, 1, ..., |Ĉ_1| − 1}.
output : σ ∈ Ĉ_1 with rank m in Ĉ_1.
/* Convert m to local ranks R[1, 2, ..., d]                */
m ← ((m − 1) mod |Ĉ_1|)
for i = 1, 2, ..., d − 1 do
    R[i] ← ((m + 1) mod M̂_{k+1})
    m ← ⌊m / M̂_{k+1}⌋
R[d] ← ((m + 1) mod k!)
/* Construct σ                                             */
π_d ← UnrankComplete(R[d])
for i ∈ W_d do
    σ[i] ← π_d[i − k(d − 1)] · d
x ← π_d[1] · d
for j = d − 1, ..., 1 do
    π_j ← UnrankAux(R[j])
    for i ∈ W_j do
        if π_j[i] = k + 1 then
            σ[i] ← x
        else
            σ[i] ← π_j[i − k(j − 1)] · d + j
    if π_j[1] ≠ k + 1 then
        x ← π_j[1] · d + j
σ[1] ← x
return σ
```

$O(n)$ steps. They also make a call to one of `RankComplete`(π), `UnrankComplete`(m) and $(d-1)$ calls to one of `RankAux`(π), `UnrankAux`(m), costing $O(k^2)$ operations each. ■

We also note in particular that in the regime $d = \Theta(n)$, we have $k = \Theta(1)$, and Theorem 38 yields linear run-time $O(n)$.

VII. Snake-in-the-box codes in S_{2m+2}

As mentioned before in Section III, the issue of asymmetry between "push-to-the-top" codes in the symmetric group of odd and even orders has also frustrated research into error-detecting codes under the Kendall τ-metric in the past.

The *Kendall τ-metric* [25] on S_n is defined as

$$d_{\mathcal{K}}(\sigma,\tau) = \left|\{(i,j) \mid \sigma(i) < \sigma(j) \wedge \tau(i) > \tau(j)\}\right| .$$

Informally, as noted in [24], it measures the minimal number of adjacent transpositions required to transform one permutation into the other, that is, the minimal r such that

$$\sigma = \tau \circ (i_1, i_1 + 1) \circ (i_2, i_2 + 1) \circ \ldots, \circ (i_r, i_r + 1)$$

for some $i_1, i_2, \ldots, i_r \in [n-1]$. An $(n, M, \mathcal{K})$-snake, or $\mathcal{K}$-snake for short, is a single-error-detecting rank-modulation Gray code of size M, or more formally, a $G_{\uparrow}(n, M)$ code C such that for all $\sigma, \tau \in C$, $\sigma \neq \tau$, it holds that $d_{\mathcal{K}}(\sigma,\tau) \geqslant 2$. Put differently, for no $i \in [n-1]$ does it hold that $\sigma = \tau \circ (i, i+1)$.

The authors have shown in [44][Thm. 17] that any $\mathcal{K}$-snake $C \subseteq S_n$ which employs a "push-to-the-top" transition on an even index $t_{\uparrow 2m}$–for any $m \in \lfloor \frac{n}{2} \rfloor$–must satisfy $|C| \leqslant \frac{n!}{2} - \Theta(n)$. Horovitz and Etzion posited in [22] that $\mathcal{K}$-snakes in S_{2m+2} do not exceed the size of those in S_{2m+1}, a conjecture refuted when Zhang and Ge demonstrated in [46] the existence of $\mathcal{K}$-snakes in S_{2m+2} of size $\frac{(2m+2)!}{4}$. Concurrently and independently, Holroyd conjectured in [20] that $\mathcal{K}$-snakes can be found in S_{2m+2} with size greater than $\frac{(2m+2)!}{2} - O(m^2)$.

A resemblance is evident in the definitions of $(n, M, \mathcal{K})$-snakes and $G_{\uparrow}^{\text{aux}}(n, M)$ codes, which is reinforced by the observations that, similarly to properties seen in Section III, any parity-preserving $G_{\uparrow}(n, M)$ code is an $(n, M, \mathcal{K})$-snake (see [44][Lem. 5]), and any $(n, M, \mathcal{K})$-snake satisfies $M \leqslant \frac{n!}{2}$ (see [44][Thm. 15]).

We wish to demonstrate how the principles behind Theorem 14 can be applied to the construction of a $\mathcal{K}$-snake in S_{2m+2} of size $M \approx \frac{(2m+2)!}{2}$.

Lemma 39 [22, Thm. 18] [45] For $m \geqslant 2$, there exist parity-preserving $G_{\uparrow}(2m+1, M_{2m+1})$ codes with

$$M_{2m+1} = |A_{2m+1}| - (2m-1) = \frac{(2m+1)!}{2} - (2m-1).$$

In particular, such a code C was constructed such that, as a group,

$$C = A_{2m+1} \setminus \left\{ t_{\uparrow 2m-1}{}^{q} \sigma \right\}_{q=0}^{2m-2}$$

for some $\sigma \in A_{2m+1}$. Finally, C only employed $t_{\uparrow 2m-1}$, $t_{\uparrow 2m+1}$.

As before, we fix $m \geqslant 2$. We also reuse

$$\begin{aligned}\varphi(\pi) &= {t_{\uparrow 2m+2}}^{2} \circ {t_{\uparrow 2m-1}}^{-1}(\pi) \\ &= \pi \circ (1, 2m+1)(2m+2, 2m, 2m-1, \dots, 2)\end{aligned}$$

and the permutations $\hat{\pi}_r = \varphi^r(\mathrm{Id})$.

Theorem 40 For all $r \geqslant 0$ a parity-preserving $G_{\uparrow}\left(2m+2, \frac{(2m+1)!}{2} - (2m-1)\right)$ code $\hat{P}_r$ exists which satisfy:

1) The first permutation in $\hat{P}_r$ is $\hat{\pi}_r$.
2) The last permutation in $\hat{P}_r$ is ${t_{\uparrow 2m-1}}^{-1}\hat{\pi}_r$.
3) For all $\pi \in \hat{P}_r$ it holds that

$$\begin{aligned}\pi(2m+2) &= \hat{\pi}_r(2m+2) \\ &= \begin{cases} 2m+2 & r \equiv 0 \pmod{2m}, \\ 2m+1-(r \bmod 2m) & r \not\equiv 0 \pmod{2m}. \end{cases}\end{aligned}$$

4) $\tilde{\sigma}_r \notin \hat{P}_r$, where we denote

$$\tilde{\sigma}_r = \left({t_{\uparrow 2m+2}}^{-1}\hat{\pi}_r\right) \circ (2m+1, 2m+2)$$

(and observe $\tilde{\sigma}_r = {t_{\uparrow 2m+1}}^{-1}(\hat{\pi}_r)$, hence in particular $\tilde{\sigma}_r(2m+2) = \hat{\pi}_r(2m+2)$).

Proof: By Lemma 39 we know that there exist a parity-preserving $G_{\uparrow}(2m+1, M_{2m+1})$ code P such that, as a set,

$$P = S_{2m+1} \setminus \left\{{t_{\uparrow 2m-1}}^{q}\sigma\right\}_{q=0}^{2m-2}$$

for some $\sigma \in A_{2m+1}$. We also know that P only employs $t_{\uparrow 2m-1}$, $t_{\uparrow 2m+1}$ transitions.

We apply its generating sequence to $\hat{\pi}_r$ to generate the $G_{\uparrow}(2m+2, M_{2m+1})$ code $\hat{P}$, which employs only $t_{\uparrow 2m-1}$, $t_{\uparrow 2m+1}$ transitions (in particular, it never employs $t_{\uparrow 2m+2}$, hence point 3 is established), and note that as a set

$$\begin{aligned}\hat{P} &= \{\tau \in A_{2m+2} \mid \tau(2m+2) = \hat{\pi}_r(2m+2)\} \\ &\quad \setminus \left\{{t_{\uparrow 2m-1}}^{q}\hat{\sigma}\right\}_{q=0}^{2m-2}\end{aligned}$$

for some $\hat{\sigma} \in A_{2m+2}$, satisfying $\hat{\sigma}(2m+2) = \hat{\pi}_r(2m+2)$.

Denote $\hat{P} = (c_j)_{j=1}^{M_{2m+1}}$. We modify our code by defining

$$\hat{P}_r = \left(c'_j\right)_{j=1}^{M_{2m+1}} = \left(\tilde{\sigma}_r \hat{\sigma}^{-1} c_j\right)_{j=1}^{M_{2m+1}},$$

which is still a $G_{\uparrow}(2m+2, M_{2m+1})$ since "push-to-the-top" transitions are group-actions by right-multiplication. Moreover, since $\tilde{\sigma}_r(2m+2) = \hat{\sigma}(2m+2) = \hat{\pi}_r(2m+2)$, as a set we have

$$\begin{aligned}\hat{P}_r &= \{\tau \in A_{2m+2} \mid \tau(2m+2) = \hat{\pi}_r(2m+2)\} \\ &\quad \setminus \left\{{t_{\uparrow 2m-1}}^{q}\tilde{\sigma}_r\right\}_{q=0}^{2m-2}.\end{aligned}$$

Note in particular that

$$\tilde{\sigma}_r(2m+1) = \hat{\pi}_r(1) \neq \hat{\pi}_r(2m+1),$$

hence $\hat{\pi}_r \in \hat{P}_r$. In addition, point 4 is thus substantiated.

Finally, ${t_{\uparrow 2m+1}}^{-1}(\hat{\pi}_r) = \tilde{\sigma}_r \notin \hat{P}_r$ implies that $\hat{\pi}_r$ must necessarily be preceded in $\hat{P}_r$ by $t_{\uparrow 2m-1}$, which substantiates point 2 (after a proper cyclic shift of $\hat{P}_r$). ■

As in Section III, $\hat{P}_r \subseteq A_{2m+2}$ for all r. We construct a $(2m+2, M, \mathcal{K})$-snake by stitching together $\hat{P}_0, \hat{P}_1, \dots, \hat{P}_{2m-1}$ in the following lemma.

Lemma 41 For all $r \geqslant 0$, we may concatenate $\hat{P}_r, \hat{P}_{r+1}$ into a (non-cyclic) "push-to-the-top" code by applying the transitions $t_{\uparrow 2m+2}, t_{\uparrow 2m+2}$ to the last permutation of $\hat{P}_r$, which is ${t_{\uparrow 2m-1}}^{-1}\hat{\pi}_r$.

The only odd permutation in the resulting code is then

$$\beta_{r+1} = {t_{\uparrow 2m+2}}^{-1}(\hat{\pi}_{r+1}),$$

which we again call the $(r+1)$-bridge.

Proof: Exactly as in the proof of Lemma 12, given that P_r, $\hat{P}_r$ are parity-preserving, and have the same first and last permutations. ■

Again, similarly to Section III, Lemma 41 can be used iteratively to cyclically concatenate $\hat{P}_0, \hat{P}_1, \dots, \hat{P}_{2m-1}$, with a single odd permutation–the r-bridge–between $\hat{P}_{(r-1) \bmod 2m}, \hat{P}_r$. Let us prove that fact in the following theorem.

Theorem 42 There exists a $(2m+2, \check{M}_{2m+2}, \mathcal{K})$-snake for all $m \geqslant 2$, with

$$\begin{aligned}\check{M}_{2m+2} &= \frac{2m}{2m+2} \cdot \frac{(2m+2)!}{2} - (2m-2)2m \\ &= \frac{2m}{2m+2} \cdot \frac{|S_{2m+2}|}{2} - (2m-2)2m.\end{aligned}$$

Proof: We define P, similarly to Section III, as the cyclic concatenation

$$\hat{P}_0, \beta_1, \hat{P}_r, \beta_2, \dots, \beta_{2m-1}, \hat{P}_{2m-1}, \beta_0.$$

Suppose $\pi_1, \pi_2 \in C$ satisfy

$$\pi_1 = \pi_2 \circ (i, i+1)$$

for some $i \in [2m+1]$, then w.l.o.g π_2 is odd and hence $\pi_2 = \beta_r$ for some $0 \leqslant r < 2m$, and π_1 is even and thus not a bridge; it must follow, then, that

$$\pi_2(2m+2) \in \{1, 2m+1\} \not\ni \pi_1(2m+2),$$

hence $i = 2m+1$ and

$$\begin{aligned}\pi_1 &= \pi_2 \circ (2m+1, 2m+2) \\ &= \left({t_{\uparrow 2m+2}}^{-1}(\hat{\pi}_r)\right) \circ (2m+1, 2m+2) \\ &= {t_{\uparrow 2m+1}}^{-1}(\hat{\pi}_r) = \tilde{\sigma}_r.\end{aligned}$$

This is in contradiction to Theorem 40, since $\pi_1(2m+2) = \hat{\pi}_r(2m+2)$ and thus $\pi_1 \in \hat{P}_r$. Hence $\hat{P}$ is a $\mathcal{K}$-snake. Now, that

$$\begin{aligned}\left|\hat{P}\right| &= 2m\left[\frac{(2m+1)!}{2} - (2m-1)\right] + 2m \\ &= \frac{2m}{2m+2} \cdot \frac{(2m+2)!}{2} - (2m-2)2m\end{aligned}$$

is trivial. ■

To conclude this section, we note that $\frac{\check{M}_{2m+2}}{|S_{2m+2}|} \xrightarrow[m\to\infty]{} \frac{1}{2}$, which is optimal. The authors are unaware of any current result achieving this. We add that, in particular, in the context of $\mathcal{K}$-snakes it is common to define the *rate* of codes as $R = \lim_{m\to\infty} \frac{\log|M_{2m+2}|}{\log|S_{2m+2}|}$ (see [44]), and we naturally observe that in our case $R = 1$ (which, again, is optimal, although $R = 1$ is also achieved by existing constructions, e.g., that of [46]).

VIII. Conclusion

In this paper we proposed a new class of codes, which we dubbed j-nontransposing, leveraging codes designed for the rank-modulation scheme under the Kendall τ-metric, which we show can be used in the construction of error-correcting codes for the ℓ_∞-metric. By doing so, we were able to construct codes that achieve better asymptotic rates than previously known constructions, while also incorporating the property of being Gray codes. As with previously known constructions, we have shown that these codes allow for linear-time encoding and decoding of noisy data.

However, there remains a gap between the best known upper-bound for code sizes (either in the general case or in the specific case of Gray codes), based on the code-anticode approach presented in [39], and achievable sizes (both known constructions and proven lower-bounds). We therefore propose that more research into upper and lower bounds on achievable code sizes is warranted.

Furthermore, much as in the case of codes designed for the Kendall τ-metric, our auxiliary construction of k-nontransposing codes in S_k has some asymmetry between the cases of even- and odd-sized congruence classes. Although mostly alleviated by Theorem 14–in particular for large k–this creates an irregularity in the slope of the graph of asymptotic rate; for rankable codes, certain regions of δ even admit a positive slope, whereby a code with a higher normalized distance also has a higher rate. We posit that, as Holroyd conjectured in [20] for $\mathcal{K}$-snakes, $2n$-nontransposing codes in S_{2n} exist with size $M > (2n)!/2 - O(n^2)$. This irregularity is especially pronounced when $2n = 6$, where we have constructed an auxiliary code of size $178 << 360 = \frac{6!}{2}$. We may note, however, that in the case of $2n = 4$, the constructed auxiliary code of size 8 can be confirmed to be optimal by a manual search.

Finally, we have presented an adaptation of the solutions discussed above to the problem of $(2n, M, \mathcal{K})$-snakes, which although not yet validating Holroyd's conjecture above, is asymptotically tight.

Acknowledgments

The authors gratefully acknowledge the three anonymous reviewers and associate editor, whose insight and meticulous suggestions helped shape this paper.

References

[1] D. J. Amalraj, N. Sundararajan, and G. Dhar, "Data structure based on Gray code encoding for graphics and image processing," in *Proccedings of the SPIE: International Society for Optical Engineering*, vol. 1349, 1990, pp. 65–76.

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

[3] T. Berger, F. Jelinek, and J. K. Wolf, "Permutation codes for sources," *IEEE Trans. on Inform. Theory*, vol. IT-18, no. 1, pp. 160–169, Jan. 1972.

[4] I. F. Blake, "Permutation codes for discrete channels," *IEEE Trans. on Inform. Theory*, vol. 20, pp. 138–140, 1974.

[5] I. F. Blake, G. Cohen, and M. Deza, "Coding with permutations," *Inform. and Control*, vol. 43, pp. 1–19, 1979.

[6] H. Chadwick and I. Reed, "The equivalence of rank permutation codes to a new class of binary codes," *IEEE Trans. on Inform. Theory*, vol. 16, no. 5, pp. 640–641, 1970.

[7] H. D. Chadwick and L. Kurz, "Rank permutation group codes based on Kendall's correlation statistic," *IEEE Trans. on Inform. Theory*, vol. IT-15, no. 2, pp. 306–315, Mar. 1969.

[8] C. C. Chang, H. Y. Chen, and C. Y. Chen, "Symbolic Gray code as a data allocation scheme for two-disc systems," *Comput. J.*, vol. 35, pp. 299–305, 1992.

[9] G. Cohen and M. Deza, "Decoding of permutation codes," in *Intl. CNRS Colloquium, July, France*, 1977.

[10] P. F. Corbett, "Rotator graphs: an efficient topology for point-to-point multiprocessor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 3, no. 5, pp. 622–626, Sep 1992.

[11] M. Deza and P. Frankl, "On maximal numbers of permutations with given maximal or minimal distance," *J. Combin. Theory Ser. A*, vol. 22, 1977.

[12] C. Ding, F.-W. Fu, T. Kløve, and V. K. Wei, "Construction of permutation arrays," *IEEE Trans. on Inform. Theory*, vol. 48, no. 4, pp. 977–980, Apr. 2002.

[13] E. En Gad, M. Langberg, M. Schwartz, and J. Bruck, "Constant-weight Gray codes for local rank modulation," *IEEE Trans. on Inform. Theory*, vol. 57, no. 11, pp. 7431–7442, Nov. 2011.

[14] ——, "Generalized Gray codes for local rank modulation," *IEEE Trans. on Inform. Theory*, vol. 59, no. 10, pp. 6664–6673, Oct. 2013.

[15] T. Etzion, "Optimal codes for correcting single errors and detecting adjacent errors," *IEEE Trans. on Inform. Theory*, vol. 38, no. 4, pp. 1357–1360, Jul 1992.

[16] V. Faber and J. W. Moore, "High-degree low-diameter interconnection networks with vertex symmetry: the directed case," Computing and Communications Division, Los Alamos National Laboratory, Los Alamos, NM, Tech. Rep. LA-UR-88-1051, 1988.

[17] C. Faloutsos, "Gray codes for partial match and range queries," *IEEE Trans. on Software Eng.*, vol. 14, pp. 1381–1393, 1988.

[18] F.-W. Fu and T. Kløve, "Two constructions of permutation arrays," *IEEE Trans. on Inform. Theory*, vol. 50, no. 5, pp. 881–883, May 2004.

[19] F. Gray, "Pulse code communication," March 1953, U.S. Patent 2632058.

[20] A. E. Holroyd, "Perfect snake-in-the-box codes for rank modulation," *IEEE Trans. on Inform. Theory*, vol. 63, no. 1, pp. 104–110, Jan 2017.

[21] S. Hood, D. Recoskie, J. Sawada, and D. Wong, "Snakes, coils, and single-track circuit codes with spread k," *Journal of Combinatorial Optimization*, vol. 30, no. 1, pp. 42–62, July 2015.

[22] M. Horovitz and T. Etzion, "Constructions of snake-in-the-box codes for rank modulation," *IEEE Trans. on Inform. Theory*, vol. 60, no. 11, pp. 7016–7025, Nov 2014.

[23] A. Jiang, R. Mateescu, M. Schwartz, and J. Bruck, "Rank modulation for flash memories," *IEEE Trans. on Inform. 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. on Inform. Theory*, vol. 56, no. 5, pp. 2112–2120, May 2010.

[25] M. Kendall and J. D. Gibbons, *Rank Correlation Methods*. Oxford University Press, NY, 1990.

[26] T. Kløve, "Spheres of permutations under the infinity norm – permutations with limited displacement," University of Bergen, Bergen, Norway, Tech. Rep. 376, Nov. 2008.

[27] T. Kløve, T.-T. Lin, S.-C. Tsai, and W.-G. Tzeng, "Permutation arrays under the Chebyshev distance," *IEEE Trans. on Inform. Theory*, vol. 56, no. 6, pp. 2611–2617, Jun. 2010.

[28] A. Mazumdar, A. Barg, and G. Zémor, "Constructions of rank modulation codes," *IEEE Trans. on Inform. Theory*, vol. 59, no. 2, pp. 1018–1029, 2013.

[29] A. Nijenhuis and H. S. Wilf, *Combinatorial algorithms for computers and calculators*, ser. Computer science and applied mathematics. New York: Academic Press, 1978.

[30] N. Papandreou, H. Pozidis, T. Mittelholzer, G. F. Close, M. Breitwisch, C. Lam, and E. Eleftheriou, "Drift-tolerant multilevel phase-change memory," in *Proceedings of the 3rd IEEE International Memory Workshop (IMW), Monterey, CA, U.S.A.*, May 2011, pp. 22–25.

[31] J. Robinson and M. Cohn, "Counting seqeuences," *IEEE Trans. on Comput.*, vol. C-30, pp. 17–23, May 1981.

[32] R. M. Roth, *Introduction to Coding Theory*. Cambridge Univ. Press, 2006.

[33] C. D. Savage, "A survey of combinatorial Gray codes," *SIAM Rev.*, vol. 39, no. 4, pp. 605–629, Dec. 1997.

[34] M. Schwartz and T. Etzion, "The structure of single-track Gray codes," *IEEE Trans. on Inform. Theory*, vol. 45, no. 7, pp. 2383–2396, Nov 1999.

[35] M. Schwartz and P. O. Vontobel, "Improved lower bounds on the size of balls over permutations with the infinity metric," to appear in IEEE Trans. on Inform. Theory.

[36] M.-Z. Shieh and S.-C. Tsai, "Decoding frequency permutation arrays under Chebyshev distance," *IEEE Trans. on Inform. Theory*, vol. 56, no. 11, pp. 5730–5737, Nov. 2010.

[37] R. C. Singleton, "Generalized snake-in-the-box codes," *IEEE Transactions on Electronic Computers*, vol. EC-15, no. 4, pp. 596–602, Aug 1966.

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

[39] I. Tamo and M. Schwartz, "Correcting limited-magnitude errors in the rank-modulation scheme," *IEEE Trans. on Inform. Theory*, vol. 56, no. 6, pp. 2551–2560, Jun. 2010.

[40] ——, "On the labeling problem of permutation group codes under the infinity metric," *IEEE Trans. on Inform. Theory*, vol. 58, no. 10, pp. 6595–6604, Oct 2012.

[41] H. Vinck, J. Haering, and T. Wadayama, "Coded M-FSK for power line communications," in *Proceedings of the 2000 IEEE International Symposium on Information Theory (ISIT2000), Sorrento, Italy*, 2000, p. 137.

[42] X. Wang and F.-W. Fu, "Constructions of snake-in-the-box codes under the ℓ_∞-metric for rank modulation," *arXiv preprint arXiv:1601.05539*, 2016.

[43] A. Williams, "The greedy Gray code algorithm," in *Algorithms and Data Structures: 13th International Symposium, WADS 2013, London, ON, Canada, August 12-14, 2013. Proceedings*, ser. Lecture Notes in Computer Science, F. Dehne, R. Solis-Oba, and J.-R. Sack, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, vol. 8037, pp. 525–536. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-40104-6_46

[44] Y. Yehezkeally and M. Schwartz, "Snake-in-the-box codes for rank modulation," *IEEE Trans. on Inform. Theory*, vol. 58, no. 8, pp. 5471–5483, Aug 2012.

[45] Y. Zhang and G. Ge, "Snake-in-the-box codes for rank modulation under Kendall's τ-metric," *IEEE Trans. on Inform. Theory*, vol. 62, no. 1, pp. 151–158, Jan. 2016.

[46] ——, "Snake-in-the-box codes for rank modulation under Kendall's τ-metric in S_{2n+2}," *IEEE Trans. on Inform. Theory*, vol. 62, no. 9, pp. 4814–4818, Sept 2016.

[47] H. Zhou, M. Schwartz, A. Jiang, and J. Bruck, "Systematic error-correcting codes for rank modulation," *IEEE Trans. on Inform. Theory*, vol. 61, no. 1, pp. 17–32, Jan. 2015.

Yonatan Yehezkeally (S'12) is a graduate student at the Department of Electrical and Computer Engineering, Ben-Gurion University of the Negev, Beer-Sheva, Israel. His research interests include algebraic coding, combinatorial structures, and finite group theory.

Yonatan received the B.Sc. and M.Sc. degrees from Ben-Gurion University of the Negev in 2012 and 2016 respectively, from the department of Mathematics and the department of Electrical and Computer Engineering.

Moshe Schwartz (M'03–SM'10) is an associate professor at the Department of Electrical and Computer Engineering, Ben-Gurion University of the Negev, Israel. His research interests include algebraic coding, combinatorial structures, and digital sequences.

Prof. Schwartz received the B.A. (*summa cum laude*), M.Sc., and Ph.D. degrees from the Technion – Israel Institute of Technology, Haifa, Israel, in 1997, 1998, and 2004 respectively, all from the Computer Science Department. He was a Fulbright post-doctoral researcher in the Department of Electrical and Computer Engineering, University of California San Diego, and a post-doctoral researcher in the Department of Electrical Engineering, California Institute of Technology. While on sabbatical 2012–2014, he was a visiting scientist at the Massachusetts Institute of Technology (MIT).

Prof. Schwartz received the 2009 IEEE Communications Society Best Paper Award in Signal Processing and Coding for Data Storage.