# Improved Rank-Modulation Codes for DNA Storage With Shotgun Sequencing

Niv Beeri and Moshe Schwartz, *Senior Member, IEEE*

***Abstract*— A common method for reading information stored in DNA molecules is shotgun sequencing. This method outputs a histogram of the frequencies of all the molecules' substrings of a given length $\ell$. To protect against noisy readings, the rank-modulation scheme encodes the information in the relative ranking of the substring frequencies, instead of their absolute values. However, the best rank-modulation codes for shotgun sequencing have low rates which are asymptotically vanishing. In this paper we propose new constructions of rank-modulation codes for shotgun sequencing. The first code construction is systematic, allowing the user to arbitrarily set the frequencies of a large subset of the substrings, which the encoder then completes to a permutation that may be realized by a DNA molecule. The construction is then improved by allowing the user to set the frequencies of additional substrings, at the cost of imposing constraints on the frequencies. The resulting codes have higher, non-vanishing rates, compared with previously known codes. As an example, for histograms of substrings of length $\ell = 2$, and an alphabet of size 4 (as in DNA molecules), we are able to construct a code with rate $\approx 0.909$, whereas previously, the best construction resulted in a code with rate $\approx 0.654$. Additionally, the encoded information in our construction may be written to shorter DNA molecules than possible before. We also prove that the systematic codes constructed in this paper are the largest possible among all systematic codes.**

***Index Terms*— DNA storage, permutation codes, De Bruijn graphs.**

## I. Introduction

Storing information in DNA molecules offers unparalleled information density, and has been proven to be feasible [6], [8], [12], [30]. Already, a density of $2.15 \cdot 10^{17}$ bytes per gram of DNA molecules has been demonstrated [9], whereas the densest commercially available option [1] is capable of storing only $1.86 \cdot 10^{11}$ bytes per gram of hardware. Long DNA sequences may be read relatively accurately using the *shotgun sequencing* technique (see [21] and the survey [20]). In this method, several copies of the same DNA sequence are broken down into fragments. These fragments are identified, and an algorithm reconstructs the DNA sequence using the knowledge of the multiset of fragments obtained. Other similar variants of this reconstruction method have also been studied [2], [10], [11], [23].

It has been suggested by [18] that we may skip the final phase of sequence reconstruction, instead opting to have the information encoded in the multiset of fragments. More precisely, if the sequence is over an alphabet $\Sigma$, the shotgun sequencing procedure provides us with a histogram, or a *profile vector*, counting how many times each $\ell$-gram (substring of length $\ell$) from $\Sigma^\ell$ (the set of all strings of length $\ell$ over the alphabet $\Sigma$) appears as a substring of the DNA sequence. Thus, the actual sequence is of no consequence, acting merely as a vehicle for its profile vector. As a side benefit, this allows us to use ambiguous profile vectors that may describe more than one sequence.

The profile vector obtained as part of the shotgun-sequencing procedure is unfortunately noisy. Errors in it are mainly due to substitution errors in the sequence-synthesis phase, non-uniform fragmentation causing coverage gaps, and $\ell$-gram substitutions due to sequencing [18]. One approach, studied in [18] is to protect the profile vector using an error-correcting code, where an appropriate metric is formulated to capture the error patterns mentioned.

Another suggestion put forth by [18], and later studied by [24], was to employ the rank-modulation scheme over the profile vectors. Rank modulation has a long history, starting with [5], [7], [25] for vector digitization and signal detection, through communication over power lines [28], and more recently, for information storage in non-volatile memories [16]. In our context, instead of storing the information in the profile vector, whose integer entries count the number of occurrences of each $\ell$-gram from $\Sigma^\ell$, the information is stored in the permutation over $\Sigma^\ell$ which is the ranking (by frequency of appearance) of the entries of the profile vector. By doing so we immediately gain a layer of protection since perturbations of the profile vector that do not result in a change of ranking, do not corrupt the stored information. Additionally, there are known error-correcting codes for the rank-modulation scheme, which we may use to gain further protection [3], [14], [15], [17], [19], [26], [31]–[35].

Not all permutations on $\Sigma^\ell$ correspond to a ranking of a profile vector of some sequence, as was observed in [24]. A linear programming algorithm was derived in [24], which can decide whether a given permutation is feasible. However, an exact characterization of all feasible permutations is still unknown. Thus, [24] provided only upper bounds on the number of feasible permutations, and recursive constructions

Manuscript received January 15, 2021; revised November 15, 2021; accepted January 29, 2022. Date of publication February 16, 2022; date of current version May 20, 2022. This work was supported in part by the Israel Science Foundation (ISF) under Grant 270/18. *(Corresponding author: Moshe Schwartz.)*

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

Communicated by L. Dolecek, Associate Editor for Coding Techniques.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TIT.2022.3152392.

Digital Object Identifier 10.1109/TIT.2022.3152392

that may also act as encoders. These constructions produce codes whose rate is asymptotically $\frac{1}{\ell}$ when $\ell$ is constant and the alphabet size $q = |\Sigma|$ goes to infinity, and 0 when $q$ is fixed and $\ell \to \infty$. Additionally, the length of the resulting encoded sequence was bounded and shown to be polynomial in $q^{\ell}$. We also note that while [18] suggested the rank-modulation scheme, it did so only for a strict subset of the entries of the profile vector.

The goal of this paper is to construct rank-modulation codes that improve upon the best known ones, namely those from [24]. Our main contributions are the following: We construct systematic codes for all alphabet sizes $q \geqslant 3$, and all window sizes $\ell \geqslant 2$. We give an efficient encoding algorithm for these codes. The asymptotic rate of these codes is 1 when $\ell$ is fixed and $q \to \infty$, and is $1 - \frac{1}{q}$ when $q$ is fixed and $\ell \to \infty$, improving upon [24]. The length of the encoded sequence is analyzed and upper bounded by $O(q^{5\ell})$ for $\ell \geqslant 3$, and $O(q^6)$ when $\ell = 2$. These improve upon the order of the corresponding bounds from [24]. Additionally, our upper bound is numerically lower than that of [24] except for the case of $q = 3$ and $\ell = 2$. We also prove an upper bound on the size of systematic codes, which shows our construction produces optimal systematic codes. Finally, we show a construction of non-systematic codes that gives codes which are strictly larger than their systematic counterparts.

The paper is organized as follows. In Section II we give the necessary definitions used throughout the paper. In Section III we construct systematic codes, provide an encoder, analyze the resulting sequence length, and prove an upper bound on the size of such codes. In Section IV we build larger codes that are non-systematic. We conclude in Section V with a summary and discussion of the results, as well as some open problems.

## II. Preliminaries

Throughout the paper we use $\Sigma$ to denote an alphabet of size $q$. We assume no further structure on the alphabet. We use $\Sigma^{\ell}$ to denote the set of all strings over $\Sigma$ of length $\ell$, also called $\ell$-grams, and $\Sigma^*$ to denote the set of all finite strings over $\Sigma$. If $s, s' \in \Sigma^*$ are strings, we use $ss'$ to denote their concatenation, and $|s|$ to denote the length of $s$. If the need arises to consider specific letters in a string $s \in \Sigma^n$, we shall usually denote the $i$th letter as $s_i$, namely, $s = s_0 s_1 \dots s_{n-1}$, where $s_i \in \Sigma$ for all $i \in [n] \triangleq \{0, 1, \dots, n-1\}$.

If $G = (V, E)$ is a directed graph, we denote the edge $e \in E$ from $v \in V$ to $v' \in V$ by $e = v \to v'$. We shall also say its source is $\mathrm{src}(e) = v$ and its destination is $\mathrm{dest}(e) = v'$. Additionally, for any vertex $v \in V$ we denote by $E_{\mathrm{in}}(v)$ the set of edges entering $v$, and similarly, we use $E_{\mathrm{out}}(v)$ to denote the set of edges leaving $v$, i.e.,

$$E_{\mathrm{in}}(v) \triangleq \{e \in E |\, \mathrm{dest}(e) = v\},$$
$$E_{\mathrm{out}}(v) \triangleq \{e \in E |\, \mathrm{src}(e) = v\}.$$

The in-degree and out-degree of $v$ are similarly defined,

$$d_{\mathrm{in}}(v) \triangleq |E_{\mathrm{in}}(v)| \qquad d_{\mathrm{out}}(v) \triangleq |E_{\mathrm{out}}(v)|.$$

These definitions are extended to sets of vertices in the natural way. Let $V' \subseteq V$ be a subset of vertices. Then we define

$$E_{\mathrm{in}}(V') \triangleq \{e \in E |\, \mathrm{dest}(e) \in V', \mathrm{src}(e) \notin V'\},$$
$$E_{\mathrm{out}}(V') \triangleq \{e \in E |\, \mathrm{src}(e) \in V', \mathrm{dest}(e) \notin V'\}.$$

### A. Strings, Profiles, and Weighted De Bruijn Graphs

A useful tool in the context of string analysis is the De Bruijn graph, which is defined as follows.

*Definition 1:* The De Bruijn graph of order $\ell \geqslant 1$ over $\Sigma$ is the directed graph $G_{q,\ell}$ whose vertex set is $V(G_{q,\ell}) = \Sigma^{\ell}$, and whose edge set is

$$E(G_{q,\ell}) = \{w_0 w_1 \dots w_{\ell-1} \to w_1 w_2 \dots w_{\ell} | \text{for all } w_i \in \Sigma\}.$$

By definition, each vertex of $G_{q,\ell}$ is identified with a string $w_0 w_1 \dots w_{\ell-1}$ from $\Sigma^{\ell}$. Furthermore, we observe that in $G_{q,\ell-1}$, the edge $w_0 w_1 \dots w_{\ell-2} \to w_1 w_2 \dots w_{\ell-1}$ is also uniquely identified by $w_0 w_1 \dots w_{\ell-1} \in \Sigma^{\ell}$. Let $s = s_0 \dots s_{n-1} \in \Sigma^n$ be a string. We say that $s_i s_{i+1} \dots s_{i+\ell-1}$ is a window of length $\ell$ into $s$, where indices are taken modulo $n$ (i.e., we consider the string cyclically). Thus, by scanning $s$ with a sliding window of length $\ell$, we obtain a cycle in $G_{q,\ell}$ whose sequence of vertices corresponds to the windows into $s$. Alternatively, with the same sliding window of length $\ell$ we obtain a cycle in $G_{q,\ell-1}$ whose sequence of edges corresponds to the windows into $s$. This latter correspondence between cycles in $G_{q,\ell-1}$ and strings will be used throughout the paper.

Motivated by the process of shotgun sequencing, previous papers [18], [24] suggested that information be encoded in the profile vector of the DNA sequence, whose definition follows shortly. First, let $A$ and $B$ be sets. We denote by $B^A$ the set of all functions from $A$ to $B$. When $A$ is finite, as shall be our case, we can think of $f \in B^A$ as vector indexed by the elements of $A$, and containing elements of $B$. Namely, in index $a \in A$ the vector contains $f(a) \in B$.

*Definition 2:* Let $s = s_0 \dots s_{n-1} \in \Sigma^n$ be a string. The *profile vector of $s$ of order $\ell$*, denoted by $p_{s,\ell} \in (\mathbb{N} \cup \{0\})^{\Sigma^{\ell}}$, is a non-negative integer vector indexed by $\Sigma^{\ell}$ such that for each $w \in \Sigma^{\ell}$, $p_{s,\ell}(w)$ counts the number of occurrences of $w$ in $s$ (cyclically). Formally,

$$p_{s,\ell}(w) \triangleq |\{i \in [n] | s_i s_{i+1} \dots s_{i+\ell-1} = w\}|,$$

where indices are taken modulo $n$.

*Definition 3:* Let $x \in (\mathbb{N} \cup \{0\})^{\Sigma^{\ell}}$. We say $x$ is *feasible* if there exists $s \in \Sigma^*$ whose profile vector of order $\ell$ is $x$, namely, $p_{s,\ell} = x$.

*Example 1:* Let $\Sigma = \{A, C, G\}$, and consider the string

$$s = GGGGAGAGAGGGGAAAAAAAACCCCCCC\ AGGGGCGCGCGCGCGCGCCCCAGCCGCCG.$$

The profile vector of $s$ of order 2 is

$$p_{s,2} = (7, 1, 5, 2, 11, 8, 4, 9, 10), \tag{1}$$

where the indices of the profile vector are in lexicographic order, i.e., $AA, AC, AG, CA, CC, CG, GA, GC, GG$.

Not every vector $x \in (\mathbb{N} \cup \{0\})^{\Sigma^{\ell}}$ is feasible. Let us build the following directed graph, $G$, with vertices $V = \Sigma^{\ell-1}$,

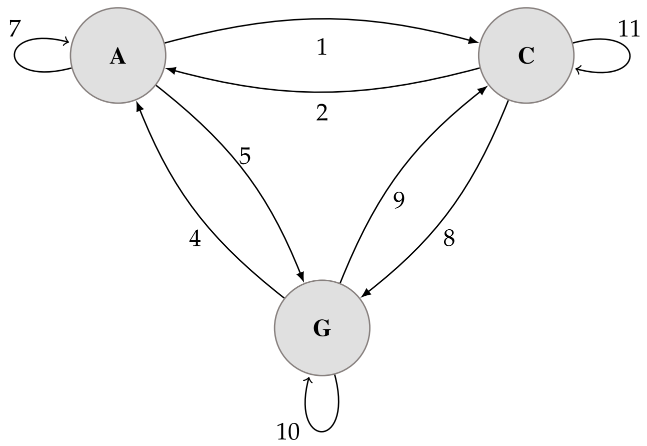

Fig. 1. The weighted De Bruijn graph of Example 2.

and for every $w = w_0 \dots w_{\ell-1} \in \Sigma^\ell$ we place $x(w)$ parallel copies of the edge $w_0 \dots w_{\ell-2} \to w_1 \dots w_{\ell-1}$. Then by our previous discussion of De Bruijn graphs, it is obvious that $x$ is the profile vector of order $\ell$ of some string $s$ if and only if $G$ contains an Eulerian cycle (i.e., a cycle passing through each edge exactly once). In turn, an Eulerian cycle exists if and only if $G$ is strongly connected (excluding isolated vertices) and for every vertex $v \in V$, its in-degree equals its out-degree, $d_{\mathrm{in}}(v) = d_{\mathrm{out}}(v)$.

For our convenience, we replace the $x(w)$ parallel edges discussed above with a single edge of weight $x(w)$. In general, for a directed graph $G = (V, E)$ we use $\mathrm{wt}_G(e) \in \mathbb{R}$ to denote the weight of an edge $e \in E$. We omit the subscript $G$ if it is clear from context. We also extend this definition to subsets of edges $E' \subseteq E$ by defining $\mathrm{wt}(E') \triangleq \sum_{e \in E'} \mathrm{wt}(e)$.

*Definition 4:* Let $G = (V, E)$ be a directed weighted graph. We say $G$ is *balanced* if $\mathrm{wt}(E_{\mathrm{in}}(v)) = \mathrm{wt}(E_{\mathrm{out}}(v))$ for all $v \in V$.

We therefore have the following corollary, translating our previous observation that uses parallel edges, to one using weights.

*Lemma 1:* A vector $x \in \mathbb{N}^{\Sigma^\ell}$ is feasible if and only if the weighted De Bruijn graph, $G_{q,\ell-1}$, with weights $\mathrm{wt}(e) = x(e)$ for all $e \in \Sigma^\ell$, is balanced.

*Proof:* Replace each edge $e$ with $\mathrm{wt}(e)$ parallel edges. Since the weight of every edge is positive, the resulting graph, $G'$, is strongly connected, and therefore $x$ is feasible if and only if $d_{\mathrm{in}}(v) = d_{\mathrm{out}}(v)$ for every $v \in G'$. But that happens if and only if $G_{q,\ell-1}$ is balanced. ■

Following [24], we shall almost always consider strings $s$ whose profile vectors are all positive integers, i.e., for all $w \in \Sigma^\ell$, $p_{s,\ell}(w) > 0$.

*Example 2:* We continue the setting of Example 1. In Fig. 1 we draw the De Bruijn graph with edge weights in accordance with the profile vector $p_{s,2}$ of (1). We observe that the resulting graph is balanced, not surprising as the profile vector was taken from a string, i.e, the profile vector is feasible.

We would like to make one more simple observation that will be useful later.

*Lemma 2:* Let $G = (V, E)$ be a finite weighted directed graph. Then $G$ is balanced if and only if for every $U \subseteq V$, $\mathrm{wt}(E_{\mathrm{in}}(U)) = \mathrm{wt}(E_{\mathrm{out}}(U))$.

*Proof:* One direction is trivial. If $\mathrm{wt}(E_{\mathrm{in}}(U)) = \mathrm{wt}(E_{\mathrm{out}}(U))$ for all $U \subseteq V$, then it is true in particular for subsets $U$ containing exactly one vertex, making $G$ balanced by definition.

In the other direction, for any $U \subseteq V$ we have

$$\begin{aligned} &\mathrm{wt}(E_{\mathrm{in}}(U)) - \mathrm{wt}(E_{\mathrm{out}}(U)) \\ &\quad = \sum_{e \in E_{\mathrm{in}}(U)} \mathrm{wt}(e) - \sum_{e \in E_{\mathrm{out}}(U)} \mathrm{wt}(e) \\ &\quad \overset{(a)}{=} \sum_{v \in U} (\mathrm{wt}(E_{\mathrm{in}}(v)) - \mathrm{wt}(E_{\mathrm{out}}(v))) = 0, \end{aligned}$$

and $(a)$ follows from the fact that edges $v' \to v''$, where $v', v'' \in U$, that are added to the sum $\sum_{v \in U} \mathrm{wt}(E_{\mathrm{in}}(v))$, are also added to the sum $\sum_{v \in U} \mathrm{wt}(E_{\mathrm{out}}(v))$, thus, canceling out. ■

## *B. Permutations and Rank Modulation*

Let $A$ be a finite set. We use $S_A$ to denote the set of permutations over $A$. Each permutation $\pi \in S_A$ may be considered as a bijection $A \to [|A|]$, sending each element of $A$ to its unique ranking in the permutation. Encoding information in permutations of the set $A$, instead of vectors over $A$, has a long history under the name *rank modulation*. The identity of the set $A$ depends on the specifics of the applications. As examples we bring [7] dealing with signal detection with impulsive noise, [28] for powerline communications, and [16] for coding in flash memories.

Recently, [18] suggested applying the rank-modulation scheme to DNA storage, with a follow-up work [24]. There, the set of permutations is $S_{q,\ell} \triangleq S_{\Sigma^\ell}$, and the ranking is done by the entries of the profile vector of the DNA sequence. Precise definitions follow:

*Definition 5:* Let $\pi \in S_{q,\ell}$ be a permutation, and let $x \in \mathbb{N}^{\Sigma^\ell}$ be some vector. We say that $x$ satisfies $\pi$, writing $x \vDash \pi$, if the entries of $x$ are distinct and for $w, w' \in \Sigma^\ell$, $\pi(w) < \pi(w')$ if and only if $x(w) < x(w')$. Additionally, we say $\pi$ is feasible if there exists a feasible $x$ that satisfies $\pi$.

We denote the set of all feasible permutations over $\Sigma^\ell$ by $\Phi_{q,\ell}$, and their number by $F_{q,\ell} \triangleq |\Phi_{q,\ell}|$. Since we will also be interested in rates, in the coding-theoretic meaning, for any non-empty subset $\mathcal{C} \subseteq S_A$, we define its *rate* as

$$R(\mathcal{C}) \triangleq \frac{\log_2 |\mathcal{C}|}{\log_2 |S_A|}.$$

We can then define the feasible rate as

$$R_{q,\ell} \triangleq R(\Phi_{q,\ell}) = \frac{\log_2 |\Phi_{q,\ell}|}{\log_2 |S_{q,\ell}|} = \frac{\log_2 F_{q,\ell}}{\log_2 (q^\ell !)}.$$

*Example 3:* We continue Example 2. When ranking the profile vector $p_{s,2}$ of (1) we obtain:

$$AC < CA < GA < AG < AA < CG < GC < GG < CC.$$

Therefore, this ranking induces a feasible permutation $\pi \in S_{3,2}$ as follows:

$$\pi = \begin{pmatrix} AA & AC & AG & CA & CC & CG & GA & GC & GG \\ 4 & 0 & 3 & 1 & 8 & 5 & 2 & 6 & 7 \end{pmatrix}, \quad (2)$$

presented in the standard two-line notation.

We shall need the following projection operator for permutations.

*Definition 6:* Let $B \subseteq A$ be two finite sets, and let $\pi \in S_A$ be a permutation over $A$. We use $\pi|_B$ to denote the unique permutation in $S_B$ that keeps the relative order of the elements of $B$ in $\pi$, namely, for all $b, b' \in B$, $\pi|_B(b) < \pi|_B(b')$ if and only if $\pi(b) < \pi(b')$.

We can think of $\pi|_B$ in the previous definition as the projection of $\pi$ onto the elements in the set $B$.

*Example 4:* We continue Example 3. Taking the permutation $\pi$ of (2), for the set $B = \{AC, CC, GA, GC\}$ we have

$$\pi|_B = \begin{pmatrix} AC & CC & GA & GC \\ 0 & 3 & 1 & 2 \end{pmatrix},$$

since $\pi(AC) < \pi(GA) < \pi(GC) < \pi(CC)$.

As usual in rank modulation, we define a code $\mathcal{C}$ to be a subset of $S_A$. If $|A| = n$ and $|\mathcal{C}| = M$, we say that $\mathcal{C}$ is an $(n, M)$-code. Of particular interest to us are systematic codes, which are analogous to systematic linear codes.

*Definition 7:* Let $A$ be some set, $|A| = n$. We say $\mathcal{C} \subseteq S_A$ is an $[n, k]$-*systematic code*, if there exists a set $B \subseteq A$, $|B| = k$, $|\mathcal{C}| = k!$, and

$$\{\pi|_B | \pi \in \mathcal{C}\} = S_B.$$

We call $B$ an *information set* for the code $\mathcal{C}$.

Intuitively, in a systematic code the user may set the ranking of the information set, $B \subseteq A$, arbitrarily (thereby, storing the user information). The remaining entries of the permutation, $A \setminus B$, are then determined by the code, creating a permutation over $A$.

## III. Optimal Systematic Codes for Feasible Permutations

In this section we study systematic codes for feasible permutation, namely, systematic subsets $\mathcal{C} \subseteq \Phi_{q,\ell}$. We provide a construction for such codes for all parameters, and show an efficient encoding algorithm. We further prove these are optimal, i.e., having the largest possible size of all systematic codes. Additionally, we analyze the length of the realizing strings, and show they are at most polynomial in the trivial lower bound.

### A. Construction

We start by giving some technical lemmas. The first shows two basic operations that take a balanced directed graph, modify the weights, but keep it balanced.

*Lemma 3:* Let $G = (V, E)$ be a finite balanced directed graph. Construct $G' = (V, E)$. Then:

1) If for all $e \in E$, $\mathrm{wt}_{G'}(e) = c \cdot \mathrm{wt}_G(e)$, where $c \in \mathbb{R}$ is some constant, then $G'$ is also balanced.
2) Let $e_0, e_1, \dots, e_{m-1}$ be a sequence of edges in $E$ that form a cycle, and let $c \in \mathbb{R}$ be some constant. If

$$\mathrm{wt}_{G'}(e) = \begin{cases} \mathrm{wt}_G(e) + c & e = e_i \text{ for some } i, \\ \mathrm{wt}_G(e) & \text{otherwise,} \end{cases}$$

then $G'$ is also balanced.

*Proof:* Multiplying the weights by a constant naturally keeps all vertices balanced. For the second case, we note that vertices that reside on the cycle have the same number of edges from the cycle entering as there are leaving. Thus, adding a constant weight to the edges of the cycle keeps the graph balanced. ■

Another simple lemma states that if we know that all but one of the vertices are balanced, then that vertex is also balanced.

*Lemma 4:* Let $G = (V, E)$ be a directed weighted graph, and let $v \in V$ be some vertex. If $\mathrm{wt}(E_{\mathrm{in}}(v')) = \mathrm{wt}(E_{\mathrm{out}}(v'))$ for all $v' \in V \setminus \{v\}$, then also $\mathrm{wt}(E_{\mathrm{in}}(v)) = \mathrm{wt}(E_{\mathrm{out}}(v))$.

*Proof:* We observe that $\{E_{\mathrm{in}}(v') | v' \in V\}$ is a partition of $E$, as is $\{E_{\mathrm{out}}(v') | v' \in V\}$. Thus,

$$\begin{aligned} \mathrm{wt}(E_{\mathrm{in}}(v)) &= \mathrm{wt}(E) - \sum_{v' \in V \setminus \{v\}} \mathrm{wt}(E_{\mathrm{in}}(v')) \\ &= \mathrm{wt}(E) - \sum_{v' \in V \setminus \{v\}} \mathrm{wt}(E_{\mathrm{out}}(v')) \\ &= \mathrm{wt}(E_{\mathrm{out}}(v)), \end{aligned}$$

which proves the claim. ■

We recall that a Hamiltonian cycle/path in a graph visits every vertex exactly once, whereas an Eulerian cycle/path visits every edge exactly once. It is well known (see [27]) that a Hamiltonian cycle in the De Bruijn graph $G_{q,\ell}$ (which is equivalent to a De Bruijn sequence) exists for all $q, \ell \geqslant 2$. Such a Hamiltonian cycle is also equivalent to an Eulerian cycle in $G_{q,\ell-1}$.

De Bruijn sequences may be nested, with lower-order sequences being prefixes of higher-order sequences. We cite the following result from [4].

*Lemma 5:* [4, Th. 1] Let $G_{q,\ell-1}$ be a De Bruijn graph with $q \geqslant 3$ and $\ell \geqslant 2$, then every Hamiltonian cycle in $G_{q,\ell-1}$ can be extended to an Eulerian cycle.

We shall further need the following technical lemma, which shows that we can complete cycles while avoiding a given Hamiltonian path.

*Lemma 6:* Let $G_{q,\ell-1}$ be a De Bruijn graph, $q \geqslant 3$, $\ell \geqslant 2$, and let $E_H = \{e_0, e_1, \dots, e_{q^{\ell-1}-1}\}$ be the edges of a Hamiltonian cycle in $G_{q,\ell-1}$. Then for any $i \in [q^{\ell-1}]$, there is a cycle passing through $e_i$ while not passing through any $e_j$, $j \neq i$.

*Proof:* As a consequence of Lemma 5, after removing the edges of $E_H$ from $G_{q,\ell-1}$, there exists an Eulerian cycle, $\beta$, in the remaining graph. Let $e_i = v_i \to v_{i+1}$ (where indices are taken modulo $q^{\ell-1}$) be the edge that we wish to complete to a cycle. Since $q \geqslant 3$, there exists at least one outgoing edge from $v_{i+1}$ and one incoming edge to $v_i$ that are not in $E_H$. Thus, at some point $\beta$ leaves $v_{i+1}$ and at some point it enters $v_i$. Denote by $\tilde{\beta}$ a part of $\beta$ that forms a path $v_{i+1} \rightsquigarrow v_i$. It now follows that $e_i, \tilde{\beta}$ is a cycle passing through $e_i$ but avoiding all $e_j$, $j \neq i$. ■

We are now ready for the main theorem of this section, that constructs a large systematic code in the space of feasible permutations. The construction steps are summarized in Algorithm 1.

*Theorem 1:* Let $G_{q,\ell-1} = (V, E)$ be a De Bruijn graph, $q \geqslant 3$, $\ell \geqslant 2$. Let $e_0, e_1, \dots, e_{q^{\ell-1}-1}$ be a sequence of edges forming a Hamiltonian cycle in $G_{q,\ell-1}$, and define $E'_H \triangleq \{e_0, \dots, e_{q^{\ell-1}-2}\}$. Then there is an injective mapping $S_{E\setminus E'_H} \to \Phi_{q,\ell}$, namely, between the set of permutations over $E \setminus E'_H$ and the set of feasible permutations over $\Sigma^\ell$.

*Proof:* Let us index the vertices of $G_{q,\ell-1}$ as $v_0, v_1, \dots, v_{q^{\ell-1}-1}$, ordered such that $e_i = v_i \to v_{i+1}$ for all $i \in [q^{\ell-1}]$ (and where indices are taken modulo $q^{\ell-1}$). We need to show that for every permutation on $E \setminus E'_H$ we can build a distinct feasible permutation on $\Sigma^\ell \cong E$.

Let $\pi \in S_{E\setminus E'_H}$ be any permutation on $E \setminus E'_H$. We assign the edges in $E \setminus E'_H$ distinct positive weights while keeping the ranking as in $\pi$. This is easily achieved by setting $\mathrm{wt}(e) = \pi(e) + 1$ for all $e \in E \setminus E'_H$. Notice that because the edges in $E'_H$ form a Hamiltonian path, then every vertex in $V \setminus \{v_{q^{\ell-1}-1}\}$ is left with exactly one outgoing edge whose weight has not been set yet.

In the next step, we assign weights to the remaining edges, i.e., the edges in $E'_H$. We do so in such a way that all vertices become balanced. For $i = 0, 1, \dots, q^{\ell-1} - 2$, in that order, we assign the weight of $e_i$ to be

$$\mathrm{wt}(e_i) = \mathrm{wt}(E_{\mathrm{in}}(v_i)) - \mathrm{wt}(E_{\mathrm{out}}(v_i) \setminus \{e_i\}).$$

Thus, all the vertices in $V \setminus \{v_{q^{\ell-1}-1}\}$ are balanced. By Lemma 4 we must have that $v_{q^{\ell-1}-1}$ is also balanced.

At this point we have assigned integer weights to all of the edges. In order for the weights to induce a permutation over $E$, we need them to be distinct. This is certainly true, by construction, for the edges in $E \setminus E'_H$. However, following the balancing process that set the weights for edges in $E'_H$, we are not guaranteed distinctness of weights for edges in $E$, and we therefore need to break ties. For the remainder of the proof we proceed with slightly different sets of edges. Define $E_H \triangleq E'_H \cup \{e_{q^{\ell-1}-1}\}$ to be the set of edges in the Hamiltonian cycle required by the theorem. Since $E \setminus E_H \subseteq E \setminus E'_H$, the weights of edges in $E \setminus E_H$ are distinct. It follows that there are only two cases in which we could be seeing equality between weights of two edges: the two edges are from $E_H$, or one edge is from $E_H$ and the other from $E \setminus E_H$.

Let us start by resolving the first case. For each $i \in [q^{\ell-1} - 1]$, let $\gamma_i$ be a cycle in $G_{q,\ell-1}$ that contains $e_i \in E_H$ but does not contain any $e \in E_H$, $e \neq e_i$. The existence of such cycles is guaranteed by Lemma 6. We define

$$\Delta \triangleq \binom{q^{\ell-1}}{2} + 1 = \frac{q^{\ell-1}(q^{\ell-1} - 1)}{2} + 1,$$

and then add $(i+1) \cdot \Delta^{-1}$ to the weight of each of the edges in $\gamma_i$, for all $i \in [q^{\ell-1} - 1]$. By Lemma 3, we therefore keep the graph balanced. We further observe that the maximum total weight added to any single edge is upper bounded by

$$\sum_{i=0}^{q^{\ell-1}-2} \frac{i+1}{\Delta} = \frac{1}{\Delta} \cdot \binom{q^{\ell-1}}{2} \leqslant 1 - \frac{1}{\Delta}. \qquad (3)$$

It follows that if $\mathrm{wt}(e) > \mathrm{wt}(e')$ before the weight addition, then this relation remains unchanged after the weight addition. In particular, the ranking of edges by weight in $E \setminus E'_H$ remains unchanged. Additionally, since distinct weights in the interval $[0, 1 - \Delta^{-1}]$ were added to the integer weights of edges of $E_H$, all weights of edges in $E_H$ are now distinct.

For the second case, we increase the weights of edges in the Hamiltonian cycle, $E_H$, by $\frac{1}{2}\Delta^{-1}$. By Lemma 3, the graph remains balanced. However, now edges in $E \setminus E_H$ have weights that are integer multiples of $\Delta^{-1}$, whereas the weights of edges in $E_H$ are not. Additionally, by (3), the addition of $\frac{1}{2}\Delta^{-1}$ to the weight does not change the ranking of edges in $E \setminus E'_H$. Thus, the second case is resolved as well.

We are now in possession of a weighted graph that is balanced, while keeping the relative ranking of weights of edges in $E \setminus E'_H$, and having distinct weights. Using Lemma 3, we now multiply all the edge weights by $2\Delta$, to obtain the same properties mentioned above, only with integer weights. Finally, we subtract $\min_{e\in E} \mathrm{wt}(e) - 1$ from all the weights. Since $G_{q,\ell-1}$ is Eulerian, by Lemma 3, the resulting weights are positive integers, and the graph has the properties mentioned above.

Let us denote the permutation induced by the weights of the edges by $\pi' \in S_E$. Clearly, by the previous discussion,

$$\pi'|_{E\setminus E'_H} = \pi,$$

hence the mapping described here, $S_{E\setminus E'_H} \to S_E$ is injective. Furthermore, since the resulting weighted graphs are all balanced, all resulting permutations are feasible and this mapping is in fact $S_{E\setminus E'_H} \to \Phi_{q,\ell}$. ■

The algorithm described in the proof of Theorem 1 is summarized using pseudocode as Algorithm 1.

*Example 5:* We demonstrate Algorithm 1 in action. Assume $\Sigma = \{A, C, G, T\}$, hence, $q = 4$. Additionally, fix the window size as $\ell = 2$. Algorithm 1 makes use of a predetermined Hamiltonian cycle, which we arbitrarily fix to be the one that is described by $AGTC$, namely, $\alpha = A \to G, G \to T, T \to C, C \to A$. Also, the algorithm requires an Eulerian cycle (whose prefix is $\alpha$), which we arbitrarily fix as the cycle described by the string $AGTCAACCTTATGGCG$ (namely, $\alpha, \beta = A \to G, G \to T, \dots$).

Assume the user supplies the following input permutation:

$$\pi = \begin{pmatrix} AA & AC & AT & CA & CC & CG \\ 9 & 0 & 8 & 1 & 12 & 4 \\ & CT & GA & GC & GG & TA & TG & TT \\ & 2 & 8 & 3 & 10 & 5 & 7 & 11 \end{pmatrix}.$$

The main steps of the algorithms are:

1) Weights taken directly from $\pi$ are assigned to edges. This is shown in Fig. 2a. The dashed edges are the Hamiltonian path, and at this point, their weight has not been determined yet.
2) Next, the algorithm determines the weights on the Hamiltonian path so that the graph becomes balanced. The result is shown in Fig. 2b. We notice that two ties form: the weight of $A \to G$ equals that of $G \to T$, and the weight of $T \to C$ equals that of $C \to G$.
3) The algorithm then proceeds to break all ties. First, ties within $\alpha$ are broken. We assume here the algorithm does no optimization when finding $s$ and $s'$ in $\beta$, and simply takes the first occurrence satisfying the requirements.

**Algorithm 1** A Systematic Encoding Algorithm for Any $q \geqslant 3$ and $\ell \geqslant 2$

```
Parameters:
  • Alphabet $\Sigma$, $|\Sigma| = q \geqslant 3$, $\ell \geqslant 2$
  • The De Bruijn graph $G_{q,\ell-1} = (V, E)$
  • A sequence of edges $\alpha = e_0, \dots, e_{q^{\ell-1}-1}$ forming a
    Hamiltonian cycle in $G_{q,\ell-1}$.
  • A sequence of edges $\beta = \tilde{e}_0, \dots, \tilde{e}_{q^\ell - q^{\ell-1}-1}$ such that
    $\alpha, \beta$ is an Eulerian cycle in $G_{q,\ell-1}$.
Input:
  • A permutation $\pi \in S_{E\setminus E'_H}$, where
    $E'_H \triangleq \{e_0, \dots, e_{q^{\ell-1}-2}\}$.
Output:
  • A profile vector $x \in \mathbb{N}^{\Sigma^\ell}$ such that $x \vDash \pi' \in S_E$ and
    $\pi'|_{E\setminus E'_H} = \pi$.
// Initial systematic part values
for $i \leftarrow 0$ to $q^\ell - q^{\ell-1} - 1$ do
    $x(\tilde{e}_i) \leftarrow \pi(\tilde{e}_i) + 1$
end
$x(e_{q^{\ell-1}-1}) \leftarrow \pi(e_{q^{\ell-1}-1}) + 1$
// Balance the graph
for $i \leftarrow 0$ to $q^{\ell-1} - 2$ do
    $x(e_i) \leftarrow \sum_{e\in E_{\text{in}}(\text{src}(e_i))} x(e) - \sum_{e\in E_{\text{out}}(\text{src}(e_i))} x(e)$
end
// Break ties in $\alpha$
$\Delta \leftarrow \binom{q^{\ell-1}}{2} + 1$
for $i \leftarrow 0$ to $q^{\ell-1} - 2$ do
    Find $s, s'$ such that $\text{src}(e_i) = \text{dest}(\tilde{e}_{s'})$ and
    $\text{dest}(e_i) = \text{src}(\tilde{e}_s)$
    for $j \leftarrow s$ to $s'$ (cyclically) do
        $x(\tilde{e}_j) \leftarrow x(\tilde{e}_j) + (i+1) \cdot \Delta^{-1}$
    end
    $x(e_i) \leftarrow x(e_i) + (i+1) \cdot \Delta^{-1}$
end
// Break ties between $\alpha$ and $\beta$
for $i \leftarrow 0$ to $q^{\ell-1} - 1$ do
    $x(e_i) \leftarrow x(e_i) + \frac{1}{2}\Delta^{-1}$
end
// Make weights integers
forall the $e \in E$ do
    $x(e) \leftarrow 2\Delta \cdot x(e)$
end
// Make weights start at 1
$m \leftarrow \min_{e\in E} x(e)$
forall the $e \in E$ do
    $x(e) \leftarrow x(e) - m + 1$
end
```

Thus, for $A \to G$ the algorithm uses $G \to G \to C \to G \to A$, for $G \to T$ it uses $T \to T \to A \to T \to G$, and for $T \to C$ it uses $C \to C \to T$. Only then ties between $\alpha$ and $\beta$ are broken. The result is shown in Fig. 2c.

4) The weights are made integers by multiplying by $2\Delta = 14$. Finally, the weights are shifted so that the minimal weight is 1, in this, reducing all weights by 13. The end result, and algorithm output, is shown in Fig. 2d.

We can see that at the end of the process we are left with weights that induce a permutation on the edges while preserving the order induced by the input permutation given by the user.

The size, and asymptotic rate of the code described in Theorem 1 is presented in the following corollary.

*Corollary 1:* For all $q \geqslant 3$ and $\ell \geqslant 2$, there exists a $[q^\ell, q^\ell - q^{\ell-1} + 1]$-systematic code $\mathcal{C}_{q,\ell} \subseteq \Phi_{q,\ell}$. Additionally, the number of feasible permutations is lower bounded by

$$F_{q,\ell} \geqslant |\mathcal{C}_{q,\ell}| = (q^\ell - q^{\ell-1} + 1)!,$$

and asymptotically the rate of feasible permutations satisfies

$$\lim_{\ell\to\infty} R_{q,\ell} \geqslant \lim_{\ell\to\infty} \frac{\log_2|\mathcal{C}_{q,\ell}|}{\log_2(q^\ell!)} = 1 - \frac{1}{q} \tag{4}$$

$$\lim_{q\to\infty} R_{q,\ell} \geqslant \lim_{q\to\infty} \frac{\log_2|\mathcal{C}_{q,\ell}|}{\log_2(q^\ell!)} = 1. \tag{5}$$

*Proof:* The existence of $\mathcal{C}_{q,\ell}$ with these parameters is immediate from Theorem 1, being the image of the injective mapping described there. For the asymptotic form, we recall Stirling's approximation, $\ln(n!) = n\ln(n) + O(n)$ (e.g., see [13, p. 452]). With that we have

$$R_{q,\ell} \geqslant \frac{\log_2|\mathcal{C}_{q,\ell}|}{\log_2(q^\ell!)} = \frac{\log_2((q^\ell - q^{\ell-1}+1)!)}{\log_2(q^\ell!)}$$
$$= \frac{(q^\ell - q^{\ell-1}+1)\log_2(q^\ell - q^{\ell-1}+1) + O(q^\ell)}{q^\ell \log_2(q^\ell) + O(q^\ell)},$$

and the claims follow. ■

At this point we compare our results with the best known ones, described in [24]. For $q \geqslant 3$ and $\ell \geqslant 2$, a non-systematic code $\mathcal{C}'_{q,\ell} \subseteq \Phi_{q,\ell}$ was constructed in [24], for which

$$|\mathcal{C}'_{q,\ell}| = 30240 \cdot \prod_{j=4}^{q}\left(j! \cdot \binom{j^2 - j + 1}{j}\right) \cdot \prod_{i=3}^{\ell} (q!)^{q^{i-1} - 2q^{i-2} + q^{i-3}}.$$

Apart for the case of $q = 3$ and $\ell = 2$ in which, in which our new code is smaller,

$$|\mathcal{C}_{3,2}| = 5040 < 30240 = |\mathcal{C}'_{3,2}|,$$

for all other cases, our new code is larger,

$$|\mathcal{C}_{q,\ell}| > |\mathcal{C}'_{q,\ell}|.$$

It should be emphasized that no explicit construction was presented in [24] for $q = 3$ and $\ell = 2$. Since this case was the basis for a recursive construction, the size 30240 was obtained via an exhaustive computer search for all feasible permutations. Asymptotically, as noted in [24],

$$\lim_{q\to\infty} \frac{\log_2|\mathcal{C}'_{q,\ell}|}{\log_2(q^\ell!)} = \frac{1}{\ell},$$

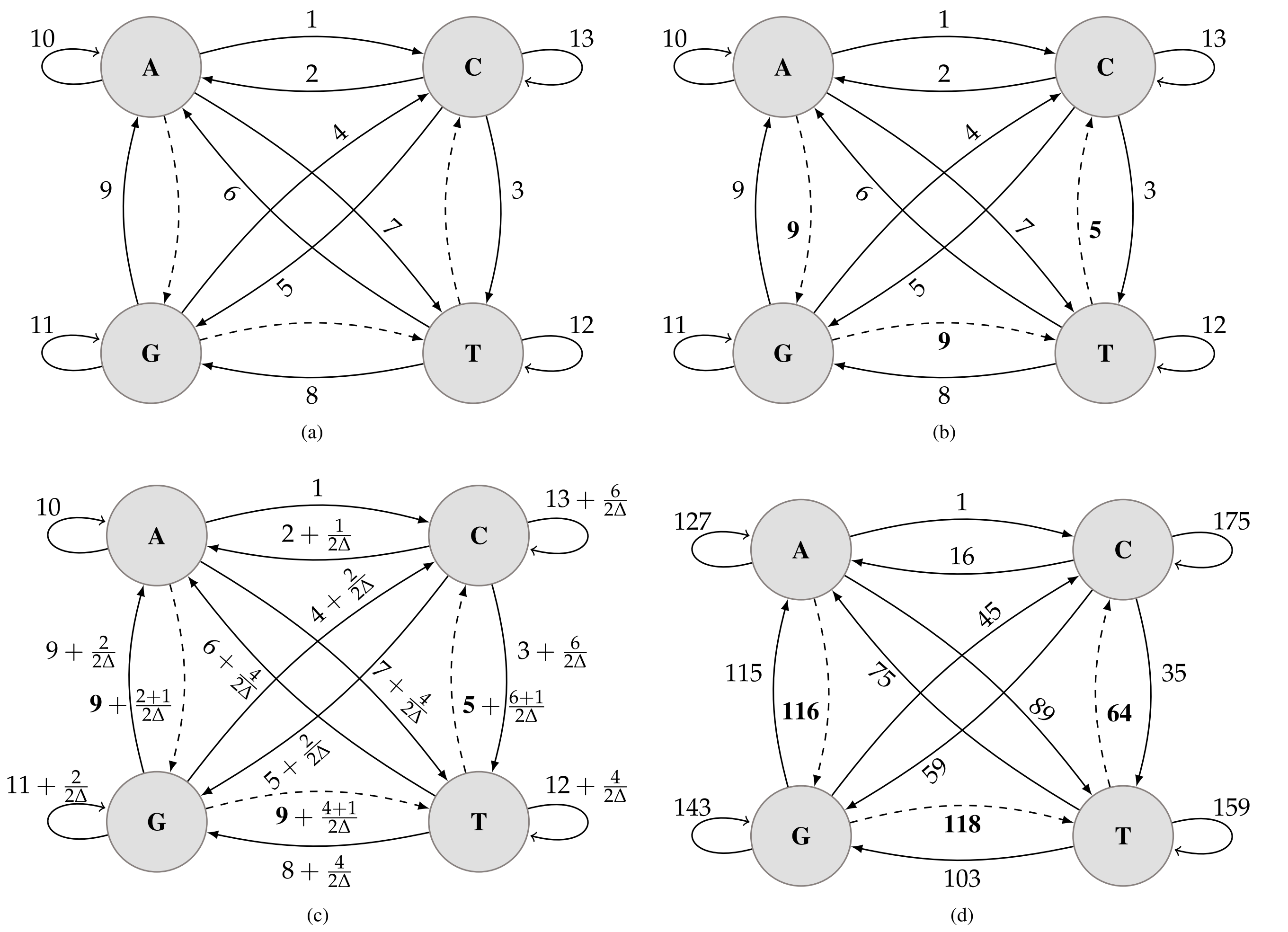

Fig. 2. A depiction of Algorithm 1 in the setting of Example 5: (a) the user information weights, (b) the initial balancing, (c) the tie breaking, and (d) the algorithm's output.

which is out-performed by our results in (5). More importantly, in practical settings $q$ is fixed while $\ell \to \infty$. In this asymptotic regime, the code of [24] gives

$$\lim_{\ell\to\infty} \frac{\log_2 |\mathcal{C}'_{q,\ell}|}{\log_2(q^\ell !)} = 0,$$

which is inferior to our results in (4) that show a non-vanishing rate.

## B. Upper Bound on the Size of Systematic Codes

Having found a construction of systematic codes for feasible permutation, it is natural to ask how large such systematic codes can be. We provide an answer in the following theorem.

*Theorem 2:* Let $q \geqslant 3$ and $k \geqslant 2$ be integers, and assume there exists a $[q^\ell, k]$-systematic code $\mathcal{C} \subseteq \Phi_{q,\ell}$. Then,

$$k \leqslant q^\ell - q^{\ell-1} + 1.$$

*Proof:* Assume to the contrary $k > q^\ell - q^{\ell-1} + 1$, and let $G_{q,\ell-1} = (V, E)$ be the De Bruijn graph and $E' \subseteq E$ be an information set of size $|E'| = k$. If we look at $G' = (V, E \setminus E')$, and forget the edge directions, then we have a graph with $q^{\ell-1}$ vertices, and strictly less than $q^{\ell-1} - 1$ edges. Since the number of edges in $E \setminus E'$ is strictly less than those required for a spanning tree, this implies that in $G'$ there exists a non-empty proper subset of vertices $\emptyset \subset V' \subset V$ that is not connected by an edge to $V \setminus V'$. Back in $G$, it then follows that

$$\begin{aligned}\emptyset &\neq E_{\mathrm{in}}(V') \triangle E_{\mathrm{out}}(V') \\ &= (E_{\mathrm{in}}(V') \setminus E_{\mathrm{out}}(V')) \cup (E_{\mathrm{out}}(V') \setminus E_{\mathrm{in}}(V')) \subseteq E',\end{aligned}$$

namely, all of the edges entering or leaving $V'$ which are not internal to $V'$, are part of the information set.

By the definition of systematic codes,

$$\{\pi|_{E'} | \pi \in \mathcal{C}\} = S_{E'}.$$

In particular, there exists $\pi \in \mathcal{C}$ such that $\pi(e) < \pi(e')$ for all $e \in E_{\mathrm{in}}(V') \setminus E_{\mathrm{out}}(V')$ and $e' \in E_{\mathrm{out}}(V') \setminus E_{\mathrm{in}}(V')$. However, $\pi$ is clearly not feasible since we cannot balance $V'$ (as is required by Lemma 2) when the weights of all externally incoming edges are smaller than the weights of all externally outgoing edges. Thus, we have reached a contradiction. ■

*Corollary 2:* The systematic codes from Theorem 1 have the largest possible size of all systematic codes in $\Phi_{q,\ell}$.

## *C. String Length*

An important figure of merit is the length of the string that the encoder generates. We would like this string to be as short as possible, to facilitate its synthesis. Thus, in this section we would like to derive an upper bound on the maximal length of the string that is generated by our algorithm. For any $q \geqslant 3$ and $\ell \geqslant 2$ we will show that this length is polynomial in the input length. A comparison with [24] will show significant improvement.

Recall that Algorithm 1 produces a profile vector, being the weights of the De Bruijn graph $G_{q,\ell-1}$. The length of the associated string is simply the total weight of all edges. We now prove an upper bound on this total weight. First, the following lemma bounds the weight of an edge on the Hamiltonian path that is used by the algorithm, before tie breaking. Since this weight might be negative, we upper bound its absolute value.

*Lemma 7:* Let $G_{q,\ell-1} = (V, E)$ be the weighted De Bruijn graph after the balancing done in Algorithm 1, but before breaking ties. Let $E'_H$ be the set of edges in $G_{q,\ell-1}$ defined in Algorithm 1, and forming a Hamiltonian path. Then for each $e \in E'_H$,

$$\begin{aligned}|\mathrm{wt}(e)| &\leqslant \mathrm{wt}(E \setminus E'_H)\\ &= \sum_{i=1}^{q^\ell - q^{\ell-1}+1} i = \binom{q^\ell - q^{\ell-1}+2}{2} \leqslant \frac{1}{2}q^{2\ell}.\end{aligned}$$

*Proof:* We use the notation of Algorithm 1. Assume $E'_H \triangleq \{e_0, \dots, e_{q^{\ell-1}-2}\}$ and $e_0, e_1, \dots, e_{q^{\ell-1}-2}$ is a Hamiltonian path in $G_{q,\ell-1}$, where $e_i$ is the edge $v_i \to v_{i+1}$. For all $i \in [q^{\ell-1}-1]$, we define $U_i \triangleq \{v_0, v_1, \dots, v_i\}$, and we observe that

$$E'_H \cap E_{\mathrm{in}}(U_i) = \emptyset, \qquad \text{and} \qquad E'_H \cap E_{\mathrm{out}}(U_i) = \{e_i\}.$$

By Lemma 2 we get that

$$\mathrm{wt}(e_i) = \mathrm{wt}(E_{\mathrm{in}}(U_i)) - \mathrm{wt}(E_{\mathrm{out}}(U_i) \setminus \{e_i\}).$$

The claim is now immediate, since both $E_{\mathrm{in}}(U_i) \subseteq E \setminus E'_H$ and $E_{\mathrm{out}}(U_i) \setminus \{e_i\} \subseteq E \setminus E'_H$. ■

*Theorem 3:* Let $G_{q,\ell-1} = (V, E)$ be the weighted De Bruijn graph that is the output of Algorithm 1. Then

$$\mathrm{wt}(E) \leqslant q^{5\ell}.$$

*Proof:* We again use the notation of Algorithm 1. Recall that all the edges in $E \setminus E'_H$ are initially given distinct weights from $\{1, \dots, q^\ell - q^{\ell-1}+1\}$, whose sum is upper bounded by $q^{2\ell}/2$, as in Lemma 7. Again, by Lemma 7, the weight of any $e \in E'_H$ satisfies $|\mathrm{wt}(e)| \leqslant q^{2\ell}/2$, before breaking ties. After the algorithm breaks all ties, the weight of each edge is increased by no more than 1. Then all weights are multiplied by $2\Delta = 2(\binom{q^{\ell-1}}{2}+1) \leqslant q^{2\ell-2}$. Finally, the normalization process may decrease or increase the weight of all edges. If an increase occurs, that it is only because some edge in $E'_H$ has negative weight. Thus, a weight of no more than $q^{2\ell}/2$ is added to all edges. It follows that the total weight of the output weighted graph satisfies,

$$\begin{aligned}\mathrm{wt}(E) &\leqslant 2\Delta\left(\frac{q^{2\ell}}{2} + (q^{\ell-1}-1)\cdot\frac{q^{2\ell}}{2} + q^\ell \cdot 1\right) + q^\ell \cdot \frac{q^{2\ell}}{2}\\ &\leqslant q^{5\ell},\end{aligned}$$

as claimed. ■

We first comment that the bound of Theorem 3 may be improved by a constant factor by having a more careful analysis in Lemma 7, taking into account the maximal cut size in $G_{q,\ell-1}$, as well as finer inequalities in Theorem 3. However, recognizing the fact that we are interested in the asymptotic regime where $q$ is constant and $\ell \to \infty$, the resulting upper bound is still $O(q^{5\ell})$.

Putting our results in context, if we denote the length of the input to Algorithm 1 by $N \triangleq q^\ell - q^{\ell-1}+1$, then the upper bound of Theorem 3 is $O(N^5)$. Thus, Algorithm 1 guarantees an output string length that is polynomial in the input length. Additionally, the absolute minimum string length is lower bounded by the case of assigning the weights $\{1, 2, \dots, q^\ell\}$ to the edges, giving a lower bound of

$$\sum_{i=1}^{q^\ell} i = \binom{q^\ell+1}{2} = \Omega(q^{2\ell}) = \Omega(N^2).$$

Finally, we would like to compare our upper bound on the length of the output string from Algorithm 1, to the upper bound on the length of the output string from the encoding algorithms in [24]. For general $q \geqslant 3$ and $\ell \geqslant 2$, it was shown in [24] that the upper bound is

$$16 \cdot \frac{2^{q-4}\cdot q! \cdot (q+1)!}{144 \cdot q^2} \cdot 3^{\ell-2} \cdot q^{\ell(q^2+1)} = O(3^\ell q^{\ell(q^2+1)}),$$

in the asymptotic regime of constant $q$ and $\ell \to \infty$. This bound is worse than that of Theorem 3.

*Remark 1:* When $\ell = 2$, the phase of tie-breaking in $\alpha$ in Algorithm 1 takes on a very simple form. This is because for every edge $e_i$, $1 \leqslant i \leqslant q^{\ell-1}-1$, the reverse edge exists in the graph, and is not part of $\alpha$. Thus, all the cycles used in this phase may be chosen to be edge disjoint, and then $\Delta$ may be reduced to $\Delta = q^{\ell-1}$. In that case, the bound on the output-string length of Theorem 3 becomes $\mathrm{wt}(E) \leqslant \frac{3}{2}q^6 + 2q^3$.

For the specific case of $\ell = 2$, [24] showed an upper bound of $q^2 \cdot 2^{q-3} \cdot \frac{q!}{6} \cdot \frac{(q+1)!}{24} \cdot 16$, which for $q = 3$ is better than the bound in Remark 1, but is otherwise worse.

## IV. Non-Systematic Codes

In the previous section we studied systematic rank-modulation codes, and we attained the maximum possible rate (Corollary 2). In this section we drop this constraint, and show that there are significantly larger codes that are non-systematic. Since the number of feasible permutations is still unknown, comparing our results with the optimum is impossible, and instead we compare against the systematic codes of the previous section.

Our first observation is a trivial increase in the code size, by using the self-loop edges in the De Bruijn graph.

*Lemma 8:* For all $q \geqslant 3$ and $\ell \geqslant 2$, there exists a $(q, M)$-code $\mathcal{C} \subseteq \Phi_{q,\ell}$, with $M = (q^\ell - q^{\ell-1}+1-q)!\frac{q^\ell!}{(q^\ell-q)!}$.

*Proof:* Remove the $q$ self-loop edges from the De Bruijn graph, and run Algorithm 1. We note that removing these edges does not affect the algorithm in any way. Then, set the weight of the self-loop edges arbitrarily. The weighted graph will remain balanced. The number of permutations obtained this way is the claimed value of $M$. ■

Next, we explore new sufficient conditions and necessary conditions for the existence of feasible permutations. We begin with a simple extension of a necessary condition presented in [24]. Since [24] used the vertices of the De Bruijn graph, whereas here we balance edge weights, we require the following new definition.

*Definition 8:* Assume $q \geqslant 3$, $\ell \geqslant 2$, and let $G_{q,\ell-1} = (V, E)$ be a weighted balanced De Bruijn graph. Let $\emptyset \subset U \subset V$ be a non-empty proper subset of $V$, and assume

$$E_{\text{in}}(U) = \{e_0, \dots, e_{k-1}\}, \qquad E_{\text{out}}(U) = \{e'_0, \dots, e'_{k-1}\},$$

are indexed such that

$$\text{wt}(e_0) < \dots < \text{wt}(e_{k-1}), \qquad \text{wt}(e'_0) < \dots < \text{wt}(e'_{k-1}).$$

We say $U$ exhibits a *Dyck configuation* if either

$$\text{wt}(e_i) < \text{wt}(e'_i) \quad \text{for all } i \in [k],$$

or

$$\text{wt}(e'_i) < \text{wt}(e_i) \quad \text{for all } i \in [k].$$

Additionally, we say a permutation $\pi \in S_{E'}$, $E_{\text{in}}(U) \cup E_{\text{out}}(U) \subseteq E' \subseteq E$, exhibits a Dyck configuration at $U$, if setting $\text{wt}(e) = \pi(e)$ for all $e \in E'$, creates a Dyck configuration at $U$.

Assume the edges in $E_{\text{in}}(U) \cup E_{\text{out}}(U) = \{e_0, \dots, e_{2k-1}\}$ are indexed such that $e_0 < e_1 < \dots < e_{2k-1}$. We can construct the following binary word $b_0, b_1, \dots, b_{2k-1}$, where $b_i$ is 0 if $e_i \in E_{\text{in}}(U)$, and is 1 otherwise. This word is a Dyck word[1] if and only if $U$ exhibits a Dyck configuration. We now present a necessary condition for a permutation to be feasible.

*Lemma 9:* Let $q \geqslant 3$ and $\ell \geqslant 2$. If $\pi \in \Phi_{q,\ell}$ is a feasible permutation, then for all $\emptyset \subset U \subset \Sigma^{\ell-1}$, $\pi$ does not exhibit a Dyck configuration at $U$.

*Proof:* Let $G_{q,\ell-1} = (V, E)$ be the De Bruijn graph, and set $\text{wt}(e) = \pi(e)$ for all $e \in E$. Assume to the contrary that $\pi$ is feasible but there exists $\emptyset \subset U \subset V = \Sigma^{\ell-1}$ that exhibits a Dyck configuration. It follows that either $\text{wt}(E_{\text{in}}(U)) < \text{wt}(E_{\text{out}}(U))$, or $\text{wt}(E_{\text{in}}(U)) > \text{wt}(E_{\text{out}}(U))$. Since $\pi$ is feasible, there exists a feasible $x \in \mathbb{N}^{\Sigma^\ell}$ such that $x \vDash \pi$. It then follows that, either

$$\sum_{e \in E_{\text{in}}(U)} x(e) < \sum_{e \in E_{\text{out}}(U)} x(e),$$

or

$$\sum_{e \in E_{\text{in}}(U)} x(e) > \sum_{e \in E_{\text{out}}(U)} x(e).$$

[1]A Dyck word is a binary sequence, exactly half of its bits are 0's, such that, in each of its prefixes, the number of 0's is at least the number of 1's.

However, the fact that $x$ is feasible implies, by Lemma 2, that

$$\sum_{e \in E_{\text{in}}(U)} x(e) = \sum_{e \in E_{\text{out}}(U)} x(e),$$

a contradiction. ■

The necessary condition for a permutation to be feasible, which was presented in Lemma 9, is unfortunately not a sufficient condition, as the following example shows.

*Example 6:* Take $q = 4$ with $\Sigma = \{A, C, G, T\}$, and $\ell = 2$. Consider the following permutation:

$$\pi = \begin{pmatrix} AA & AC & AG & AT & CA & CC & CG & CT \\ 12 & 0 & 1 & 5 & 4 & 13 & 11 & 7 \\ GA & GC & GG & GT & TA & TC & TG & TT \\ 3 & 10 & 14 & 6 & 2 & 8 & 9 & 15 \end{pmatrix}.$$

By inspection, one can verify that no $\emptyset \subset U \subset \Sigma$ exhibits a Dyck configuration. However, by computer we find that this permutation is infeasible (see the linear-programming method for deciding feasibility in [24, Section IV]).

Not all is lost though. In the next theorem we show that, compared with the systematic code of Theorem 1, the user may set another edge, provided that a Dyck configuration does not appear. To prove this claim we first bring some necessary definitions and preliminary results.

*Definition 9:* Let $G = (V, E)$ be a finite directed weighted graph. A vertex $v \in V$ is said to be in an *over* (respectively, *under*) state, if $\text{wt}(E_{\text{in}}(v)) < \text{wt}(E_{\text{out}}(v))$ (respectively, $\text{wt}(E_{\text{in}}(v)) > \text{wt}(E_{\text{out}}(v))$). Otherwise, $v$ is said to be *balanced*.

*Definition 10:* Let $G = (V, E)$ be a finite directed weighted graph. For any $e \in E$, define

$$E_{\geqslant e} \triangleq \{e' \in E | \, \text{wt}(e') \geqslant \text{wt}(e)\}.$$

We say $e^* \in E$ is a *step-up edge for* $v \in V$ if $|E_{\geqslant e^*} \cap E_{\text{in}}(v)| < |E_{\geqslant e^*} \cap E_{\text{out}}(v)|$. We say $e^* \in E$ is a *step-down edge for* $v \in V$ if $|E_{\geqslant e^*} \cap E_{\text{in}}(v)| > |E_{\geqslant e^*} \cap E_{\text{out}}(v)|$. Finally, we say $e^* \in E$ is a *stable edge for* $v \in V$ if $|E_{\geqslant e^*} \cap E_{\text{in}}(v)| = |E_{\geqslant e^*} \cap E_{\text{out}}(v)|$.

Unlike Lemma 3, we introduce an operation that may change the balanced state of vertices.

*Lemma 10:* Let $G = (V, E)$ be a finite directed weighted graph. Assume that $v \in V$ is in an over state (resp., in an under state), and that $e^* \in E$ is a step-down edge (resp., step-up edge) for $v$. Construct $G' = (V, E)$, and set its edge weights as follows:

$$\text{wt}_{G'}(e) = \begin{cases} \text{wt}_G(e) & e \notin E_{\geqslant e^*}, \\ \text{wt}_G(e) + c & e \in E_{\geqslant e^*}, \end{cases}$$

for all $e \in E$, and where

$$c = \frac{|\text{wt}_G(E_{\text{in}}(v)) - \text{wt}_G(E_{\text{out}}(v))|}{||E_{\geqslant e^*} \cap E_{\text{in}}(v)| - |E_{\geqslant e^*} \cap E_{\text{out}}(v)||}.$$

Then the relative order of edges (by weight) in $G$ and $G'$ are the same, and $v$ is balanced in $G'$.

*Proof:* The fact that the relative order of edges does not change between $G$ and $G'$, is trivial. Assume $v$ is in an under state, i.e., $\text{wt}_G(E_{\text{in}}(v)) > \text{wt}_G(E_{\text{out}}(v))$. Since $e^*$ is a step-up edge for $v$, by definition we have $|E_{\geqslant e^*} \cap E_{\text{in}}(v)| < |E_{\geqslant e^*} \cap E_{\text{out}}(v)|$. We now note that increasing the weights of the edges in $E_{\geqslant e^*}$ by 1, adds $|E_{\geqslant e^*} \cap E_{\text{in}}(v)|$ weight to the incoming edges of $v$, and $|E_{\geqslant e^*} \cap E_{\text{out}}(v)|$ weight to the outgoing edges of $v$. Thus,

$$\begin{aligned}\text{wt}_{G'}(E_{\text{in}}(v)) - \text{wt}_{G'}(E_{\text{out}}(v))\\ &= \text{wt}_G(E_{\text{in}}(v)) + c\,|E_{\geqslant e^*} \cap E_{\text{in}}(v)|\\ &\quad - \text{wt}_G(E_{\text{out}}(v)) - c\,|E_{\geqslant e^*} \cap E_{\text{out}}(v)| = 0.\end{aligned}$$

A symmetric argument proves the case when $v$ is in an over state. ■

We are now in a position to show how another edge may be set (compared with systematic codes), provided a Dyck configuration is avoided.

*Theorem 4:* Assume the same setting as in Theorem 1. Then every permutation $\pi$ on $E \setminus E'_H \cup \{e_0\}$, that does not exhibit a Dyck configuration at $\{v_1\}$, can be extended to a feasible permutation $\pi' \in \Phi_{q,\ell}$, namely, $\pi'|_{E\setminus E'_H \cup\{e_0\}} = \pi$.

*Proof:* Our goal is to show that we can find weights $x(e)$ for each $e \in E$, such that the graph is balanced, and the relative order (by weight) of the edges in $E \setminus E'_H \cup \{e_0\}$ is preserved. We start by setting $x(e) = \pi(e)+1$ for each $e \in E\setminus E'_H \cup \{e_0\}$. We then note $v_0$ is the only vertex all of whose incident edge weight have already been set.

If $v_0$ is not balanced, then it is either in an over state or an under state. Let us assume that $v_0$ is in an under state. The proof for the over state is symmetric. Arrange the edges of $E_{\text{in}}(v_0) \cup E_{\text{out}}(v_0)$ in ascending weight order, $e'_0, \dots, e'_{2k-1}$, where we note that by definition, self loops are not included in this union. Create the binary word $b_0, \dots, b_{2k-1}$, $b_i \in \{0, 1\}$, where $b_i = 0$ if and only if $e'_i \in E_{\text{in}}(v_0)$. In this word exactly half of the bits are 0's. Since there is no Dyck configuration at $\{v_0\}$, there exists a proper prefix $b_0, \dots, b_{t-1}$ that contains strictly more 0's than 1's, and therefore, $b_t, \dots, b_{2k-1}$ contains strictly more 1's than 0's. Thus, $e'_t$ is a step-up edge for $v_0$.

Using Lemma 10, we can adjust the weights of edges (that have already been assigned) and ensure that $v_0$ is balanced, while keeping the relative order of edges by weight. We also point out that the resulting weights must all be distinct. If needed, we multiply all edge weights by the same constant to obtain integer weights. We now continue by running Algorithm 1, starting from the balancing part. The resulting weights induce a permutation $\pi' \in \Phi_{q,\ell}$, as desired. ■

*Corollary 3:* For all $q \geqslant 3$ and $\ell \geqslant 2$, there exists a code $\mathcal{C}'_{q,\ell} \subseteq \Phi_{q,\ell}$ with

$$F_{q,\ell} \geqslant |\mathcal{C}'_{q,\ell}| = (q^\ell - q^{\ell-1} + 2 - q)! \cdot \frac{q-1}{q+1} \cdot \frac{q^\ell!}{(q^\ell - q)!}.$$

*Proof:* We choose $v_0$ in the setting of Theorem 4 to be a vertex with no self loops. It follows that $|E_{\text{in}}(v_0)| = |E_{\text{out}}(v_0)| = q$. We first look locally at $v_0$. We can arrange the $q$ incoming edges among themselves in $q!$ ways, and similarly for the $q$ outgoing edges. Next, we count the number of ways these two orderings may be merged so as not to exhibit a Dyck configuration. A Dyck configuration is equivalent to a Dyck word, and the number of those is known to be the Catalan number $C_q \triangleq \frac{1}{q+1}\binom{2q}{q}$ (e.g., see [13, p. 358]). There are also two ways to choose whether the first edge is from $E_{\text{in}}(v_0)$ or $E_{\text{out}}(v_0)$. We obtain that the total ways of ordering $E_{\text{in}}(v_0) \cup E_{\text{out}}(v_0)$ is given by

$$(q!)^2\left(\binom{2q}{q} - 2C_q\right) = (2q)! \cdot \frac{q-1}{q+1}.$$

We then extend this to a permutation of $E \setminus E'_H \cup \{e_0\}$, for a total number of permutations equalling

$$(q^\ell - q^{\ell-1} + 2)! \cdot \frac{q-1}{q+1}.$$

By Theorem 4 these may be injectively extended to feasible permutations of $E$. As a final step, we may employ the same strategy as Lemma 8: exclude the self loops from the entire process, and set their value only at the end. This results in the claimed number of permutations in $\mathcal{C}'_{q,\ell}$. ■

We can further improve Theorem 4, by considering permutations on *all* the edges of the De Bruijn graph. A sufficient condition is described in the following theorem.

*Theorem 5:* Assume $q \geqslant 3$, $\ell \geqslant 2$, and let $G_{q,\ell-1} = (V, E)$ be the De Bruijn graph. Let $\pi \in S_E$, and assign $\text{wt}(e) = \pi(e) + 1$, for all $e \in E$. If we can index the vertices $V = \{v_0, \dots, v_{q^{\ell-1}-1}\}$ such that for each $i \in [q^{\ell-1} - 1]$:

1) there is no Dyck configuration at $\{v_i\}$, and
2) $v_i$ has a step-up edge $e \in E$, and a step-down edge $e' \in E$, such that $e$ and $e'$ are stable edges for $v_j$, for all $j \in [i]$,

then $\pi$ is feasible, i.e., $\pi \in \Phi_{q,\ell}$.

*Proof:* When the conditions in the lemma are satisfied, we can balance each $v_0, \dots, v_{q^{\ell-1}-2}$, one by one, in this order, using Lemma 10. Note that while we balance $v_i$, we do not harm the balance of $v_0, \dots, v_{i-1}$. Also note that vertex $v_{q^{\ell-1}-1}$ is automatically balanced once all the previous ones are. The result is a balanced graph with rational weights. Multiplying all the weights by an appropriate constant we achieve a balanced graph with distinct integer positive weights, that realize the permutation $\pi$. ■

Alas, the sufficient condition for a permutation to be feasible, which was presented in Theorem 5, is not necessary, as the following example shows.

*Example 7:* Take $q = 4$ with $\Sigma = \{A, C, G, T\}$, and $\ell = 2$. Consider the following permutation:

$$\pi = \begin{pmatrix} AA & AC & AG & AT & CA & CC & CG & CT \\ 12 & 0 & 1 & 7 & 2 & 13 & 6 & 8 \\ GA & GC & GG & GT & TA & TC & TG & TT \\ 3 & 5 & 14 & 10 & 4 & 11 & 9 & 15 \end{pmatrix},$$

By inspection one can verify that the requirements of Theorem 5 are not satisfied, yet the permutation is feasible, as Fig. 3 shows.

Table I shows a comparison between the size of the codes resulting from the different methods in this paper and in [24]. We first note that the last row, the total number of feasible permutations, was obtained using an exhaustive computer search, and hence the limitation to $\ell = 2$ and $q = 3, 4$. We also observe that the entry for $q = 3$, $\ell = 2$, from [24] was obtained in the same way, i.e., an exhaustive computer search, whose results bootstrapped a recursive construction in [24].

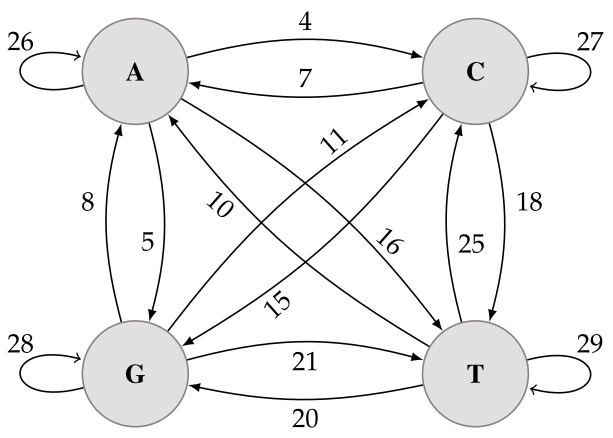

Fig. 3. The balanced graph for the permutation from Example 7.

TABLE I

THE RATE OF RANK-MODULATION CODES FOR DNA STORAGE WITH SHOTGUN SEQUENCING (IN PARENTHESES, THE CODE RATES)

| Source | $\ell = 2, q = 3$ | $\ell = 2, q = 4$ |
|---|---|---|
| [24] | 30240 ($\approx 0.806$) | 518918400 ($\approx 0.654$) |
| Theorem 1 | 5040 ($\approx 0.666$) | 6227020800 ($\approx 0.735$) |
| Theorem 4 | 30240 ($\approx 0.806$) | 95103590400 ($\approx 0.824$) |
| Theorem 5 | 30240 ($\approx 0.806$) | 1296453150720 ($\approx 0.909$) |
| Feasible permutations | 30240 ($\approx 0.806$) | 1540034496000 ($\approx 0.915$) |

## V. CONCLUSION

In this paper we studied rank-modulation codes for DNA storage when used in conjunction with shotgun sequencing. We constructed systematic codes for all parameters $q \geqslant 3$ and $\ell \geqslant 2$, which we proved are optimal. These improve upon the results of [24] by obtaining an asymptotic rate of $1 - \frac{1}{q}$ when $q$ is fixed and $\ell \to \infty$, compared with an asymptotic rate of 0 in [24]. In the asymptotic regime of $\ell$ fixed and $q \to \infty$ we obtain asymptotic rate of 1 compared with $\frac{1}{\ell}$ in [24]. The construction we gave was accompanied by an efficient encoding algorithm. We further note that the decoding process is also efficient since the permutation given to the algorithm as input is simply the ranking of the frequencies of the information edges. Finally, we also showed how larger codes may be obtained by avoiding Dyck configurations.

We would like to further discuss additional aspects that may be readily combined into the coding schemes we presented in this paper:

*a) Weight balancing:* When considering data storage in synthesized DNA molecules, it has been argued that an overall GC-content[2] of roughly $50\%$ contributes to the stability of the molecule [29]. We can adjust Algorithm 1 to accomplish this by removing the self loops from the set of information edges in the De Bruijn graph $G_{q,\ell-1}$. After running the algorithm and obtaining an encoded sequence, we may increase the weights of the relevant self loops to reach the desired GC-content of the encoded sequence.

*b) Forbidden $\ell$-grams:* Research suggests that some $\ell$-grams are likely to cause sequencing errors [22]. We can ensure these $\ell$-grams never appear as a substring of the encoded output sequence by removing their corresponding edges from the De Bruijn graph $G_{q,\ell-1}$ to obtain a graph $G'$. A careful reading of Theorem 1 and Algorithm 1 reveals that the claims hold for $G'$ as well, provided the following hold:

- $G'$ has an Eulerian cycle.
- $G'$ has a Hamiltonian path.
- For each edge $e$ on the Hamiltonian path, there is a directed cycle passing through $e$ and not through any of the other edges on the Hamiltonian path.

When these requirements hold, the edges not on the Hamiltonian path form an information set, and trivial adjustments to Algorithm 1 make it work for $G'$ as well.

*c) Error correction:* As mentioned in the introduction, the mere use of the rank-modulation scheme already protects against perturbations of the profile vector that do not change the ranking. If we desire more error-protection capabilities, then we may use any of the rank-modulation error-correcting codes known in the literature. These may be trivially combined with the systematic encoding of Section III. In the notation of Theorem 1, if $C \subseteq S_{E \setminus E'_H}$ is a rank-modulation code, then each of its codewords may be mapped to $\Phi_{q,\ell}$. The reverse process is easily accomplished by projecting the receiving permutation onto $E \setminus E'_H$, and then decoding using $C$. We can also use the larger non-systematic codes from Section IV. Assume $C_{\mathrm{out}} \subseteq S_{q,\ell}$ is the (non-systematic) code from Section IV, and let $C_{\mathrm{in}} \subseteq S_{q,\ell}$ be a rank-modulation error-correcting code. If $C_{\mathrm{in}}$ is a subgroup code (e.g., the codes studied in [26]), then its cosets partition $S_{q,\ell}$ into error-correcting codes (in the case of [26], due to the right-invariance of the $\ell_\infty$-metric on permutations). Thus, one of these cosets intersects $C_{\mathrm{out}}$ in a code that is both feasible, has the error-correction capabilities of $C_{\mathrm{in}}$, and whose size is at least $|C_{\mathrm{out}}| \cdot |C_{\mathrm{in}}| / |S_{q,\ell}|$.

We would like to mention some open questions. Finding the exact number of feasible permutations is first and foremost. The upper bound on the asymptotic rate of feasible permutations is still 1 (see [24]), whereas the lower bound has been improved in this paper to $1 - \frac{1}{q}$, assuming $q$ is constant and $\ell \to \infty$. This lower bound is obtained by considering systematic codes, and it is the best possible. Unfortunately, even though the non-systematic codes of Corollary 3 have strictly larger size compared with systematic codes, they do not offer any improvement asymptotically.

Another interesting open question concerns the length of the encoded sequences. The trivial lower bound is $\Omega(q^{2\ell})$, whereas the upper bound from the systematic codes of Section III is $O(q^{5\ell})$. What are the worst-case bound and the average-case bound is still unknown.

Yet another open problem is determining the minimum distance of feasible permutations. Several metrics have been studied in connection with rank-modulation codes, e.g., Kendall's $\tau$-metric, the $\ell_\infty$-metric (also known as Chebyshev's metric), and Ulam's metric, to name a few. Intrinsically, the set of feasible permutations may possess sufficient minimal distance to allow error correction. What this distance is, or bounds on it, are as of yet, unknown.

Finally, finding a concise sufficient and necessary condition for a permutation to be feasible, remains an open problem.

[2]The GC-content of a DNA molecule is the percentage of bases that are either $G$ or $C$.

Finding such a condition might pave the way to constructing encoders for feasible permutations. We leave all of these open problems for future work.

## REFERENCES

[1] (2020). *Nimbus Data ExaDrive DC Datasheet*. [Online]. Available: https://nimbusdata.com/docs/ExaDrive-DC-Datasheet.pdf

[2] J. Acharya, H. Das, O. Milenkovic, A. Orlitsky, and S. Pan, "String reconstruction from substring compositions," *SIAM J. Discrete Math.*, vol. 29, no. 3, pp. 1340–1371, 2015.

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

[4] V. Becher and P. A. Heiber, "On extending de Bruijn sequences," *Inf. Process. Lett.*, vol. 111, no. 18, pp. 930–932, Sep. 2011.

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

[6] J. Bornholt, R. Lopez, D. M. Carmean, L. Ceze, G. Seelig, and K. Strauss, "A DNA-based archival storage system," *ACM SIGOPS Operat. Syst. Rev.*, vol. 50, no. 2, pp. 637–649, 2016.

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

[8] G. M. Church, Y. Gao, and S. Kosuri, "Next-generation digital information storage in DNA," *Science*, vol. 337, no. 6102, p. 1628, 2012.

[9] Y. Erlich and D. Zielinski, "DNA fountain enables a robust and efficient storage architecture," *Science*, vol. 355, no. 6328, pp. 950–954, Mar. 2017.

[10] R. Gabrys and O. Milenkovic, "Unique reconstruction of coded strings from multiset substring spectra," *IEEE Trans. Inf. Theory*, vol. 65, no. 12, pp. 7682–7696, Dec. 2019.

[11] R. Gabrys, S. Pattabiraman, and O. Milenkovic, "Mass error-correction codes for polymer-based data storage," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Los Angeles, CA, USA, Jun. 2020, pp. 25–30.

[12] N. Goldman *et al.*, "Towards practical, high-capacity, low-maintenance information storage in synthesized DNA," *Nature*, vol. 494, pp. 77–80, Jan. 2013.

[13] R. L. Graham, D. E. Knuth, and O. Patashnik, *Concrete Mathematics: A Foundation for Computer Science*. Reading, MA, USA: Addison-Wesley, 1994.

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

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

[16] A. Jiang, R. Mateescu, M. Schwartz, and J. Bruck, "Rank modulation for flash memories," *IEEE Trans. Inf. Theory*, vol. 55, no. 6, pp. 2659–2673, Jun. 2009.

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

[18] H. M. Kiah, G. J. Puleo, and O. Milenkovic, "Codes for DNA sequence profiles," *IEEE Trans. Inf. Theory*, vol. 62, no. 6, pp. 3125–3146, Jun. 2016.

[19] A. Mazumdar, A. Barg, and G. Zemor, "Constructions of rank modulation codes," *IEEE Trans. Inf. Theory*, vol. 59, no. 2, pp. 1018–1029, Feb. 2013.

[20] J. R. Miller, S. Koren, and G. Sutton, "Assembly algorithms for next-generation sequencing data," *Genomics*, vol. 95, no. 6, pp. 315–327, 2010.

[21] A. Motahari, G. Bresler, and D. Tse, "Information theory for DNA sequencing: Part I: A basic model," in *Proc. IEEE Int. Symp. Inf. Theory*, Cambridge, MA, USA, Jul. 2012, pp. 2741–2745.

[22] K. Nakamura *et al.*, "Sequence-specific error profile of illumina sequencers," *Nucleic Acids Res.*, vol. 39, no. 13, p. e90, Jul. 2011.

[23] S. Pattabiraman, R. Gabrys, and O. Milenkovic, "Reconstruction and error-correction codes for polymer-based data storage," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Visby, Sweden, Aug. 2019, pp. 1–5.

[24] N. Raviv, M. Schwartz, and E. Yaakobi, "Rank-modulation codes for DNA storage with shotgun sequencing," *IEEE Trans. Inf. Theory*, vol. 65, no. 1, pp. 50–64, Jun. 2019.

[25] D. Slepian, "Permutation modulation," *Proc. IEEE*, vol. 53, no. 3, pp. 228–236, Mar. 1965.

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

[27] T. van Aardenne-Ehrenfest and N. G. de Bruijn, "Circuits and trees in oriented linear graphs," *Simon Stevin, Wis- Natuurkundig Tijdschrift*, vol. 28, pp. 203–217, Jan. 1951.

[28] H. Vinck, J. Haering, and T. Wadayama, "Coded M-FSK for power line communications," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Sorrento, Italy, Jun. 2000, p. 137.

[29] P. Yakovchuk, "Base-stacking and base-pairing contributions into thermal stability of the DNA double helix," *Nucleic Acids Res.*, vol. 34, no. 2, pp. 564–574, Jan. 2006.

[30] S. M. H. T. Yazdi, Y. Yuan, J. Ma, H. Zhao, and O. Milenkovic, "A rewritable, random-access DNA-based storage system," *Sci. Rep.*, vol. 5, no. 1, pp. 1–10, Nov. 2015.

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

[32] Y. Yehezkeally and M. Schwartz, "Limited-magnitude error-correcting Gray codes for rank modulation," *IEEE Trans. Inf. Theory*, vol. 63, no. 9, pp. 5774–5792, Sep. 2017.

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

[34] Y. Zhang and G. Ge, "Snake-in-the-box codes for rank modulation under Kendall's $\tau$-metric in $\mathrm{S}_{2n+2}$," *IEEE Trans. Inf. Theory*, vol. 62, no. 9, pp. 4814–4818, Sep. 2016.

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

**Niv Beeri** received the B.Sc. degree in mathematics, the B.Sc. degree in electrical engineering, and the M.Sc. degree in electrical engineering from the Ben-Gurion University of the Negev, Beer-Sheva, Israel. He was a Graduate Student at the School of Electrical and Computer Engineering, Ben-Gurion University of the Negev. His research interests include coding for DNA storage, graph theory, and combinatorics.

**Moshe Schwartz** (Senior Member, IEEE) received the B.A. *(summa cum laude)*, M.Sc., and Ph.D. degrees from the Computer Science Department, Technion—Israel Institute of Technology, Haifa, Israel, in 1997, 1998, and 2004, respectively.

He was a Fulbright Post-Doctoral Researcher with the Department of Electrical and Computer Engineering, University of California San Diego, and a Post-Doctoral Researcher with 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). He is currently a Professor with the School 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 2009 IEEE Communications Society Best Paper Award in Signal Processing and Coding for Data Storage and the 2020 NVMW Persistent Impact Prize. He served as an Associate Editor of coding techniques and coding theory for the IEEE TRANSACTIONS ON INFORMATION THEORY (2014–2021). Since 2021, he has been serving as an Area Editor of coding and decoding for the IEEE TRANSACTIONS ON INFORMATION THEORY. Since 2021, he has been an Editorial Board Member for the *Journal of Combinatorial Theory, Series A*.