Rank Modulation Codes for DNA Storage

Netanel Raviv[⋆,†], **Moshe Schwartz**[†], and **Eitan Yaakobi**[⋆]
[⋆]Computer Science Department, Technion – Israel Institute of Technology, Haifa 3200003, Israel
[†]Electrical and Computer Engineering, Ben-Gurion University of the Negev, Beer Sheva 8410501, Israel
netanel.raviv@gmail.com, schwartz@ee.bgu.ac.il, yaakobi@cs.technion.ac.il

***Abstract*—Synthesis of DNA molecules offers unprecedented advances in storage technology. Yet, the microscopic world in which these molecules reside induces error patterns that are fundamentally different from their digital counterparts. Hence, to maintain reliability in reading and writing, new coding schemes must be developed.**

In a reading technique called shotgun sequencing, a long DNA string is read in a sliding window fashion, and a profile vector is produced. It was recently suggested by Kiah et al. that such a vector can represent the permutation which is induced by its entries, and hence a rank modulation scheme arises. Although this interpretation suggests high error tolerance, it is unclear which permutations are feasible, and how to produce a DNA string whose profile vector induces a given permutation.

In this paper, by observing some necessary conditions, an upper bound for the number of feasible permutations is given. Further, a technique for deciding the feasibility of a permutation is devised. By using this technique, an algorithm for producing a considerable number of feasible permutations is given, which applies to any alphabet size and any window length.

I. Introduction

Coding for DNA storage devices has gained increasing attention lately, following a proof of concept by several promising prototypes [1], [3], [4]. Due to the high cost and technical limitations of the synthesis (i.e., writing) process, these works focused on producing short strings, but as the cost of synthesis declines, longer strings can be produced. In turn, long strings are read more accurately by a technique called *shotgun sequencing* [12], in which short substrings are read separately and reassembled together to form the original string.

The shotgun sequencing technique has motivated the definition of the DNA storage channel [8]. In a channel with alphabet Σ of size q and a window length ℓ, data is stored as a long string over Σ; the output of the channel is a (possibly erroneous) *histogram*, or a *profile vector* with q^ℓ entries, each containing the number of times that the corresponding ℓ-substring was observed. Errors in this channel might occur as a result of substitution errors in the synthesis or sequencing processes, or imperfect coverage of reads.

In order to cope with different error patterns, several code constructions were recently suggested [8], however, much is yet to be done to obtain high error resilience and high rate. It was also suggested in [8, Sec. VIII.B] to employ a rank modulation scheme, which has the potential for coping with *any* error pattern that does not revert the order among the entries of the profile vector.

In this scheme, the absolute values of the profile vector are ignored, and only their relative values are considered. That is, a given profile vector represents the permutation which is induced by its entries. Clearly, to induce a permutation the entries of the vector must be distinct, which is a reasonable assumption for long strings and short read length.

A well known tool in the analysis of strings is the DeBruijn graph G_q^ℓ, whose set of nodes is Σ^ℓ, and two nodes are connected by a directed edge if the $(\ell-1)$-suffix of the former is the $(\ell-1)$-prefix of the latter. This graph is a useful tool in the analysis of the DNA storage channel since any string over Σ induces a path in the graph, and since a (normalized) profile vector can be seen as measure on its node set. Further, we may restrict out attention to closed strings only (i.e., strings that correspond to closed paths), since in our context, they are asymptotically equivalent to their ordinary counterparts.

Due to flow conservation constraints in the DeBruijn graph [7], it is evident that not every permutation is feasible, i.e., there exist permutations that are not induced by any profile vector, and hence by any string. Consequently, [8] suggested to disregard a certain subset S of the entries in the profile vector, encode any permutation on the complement of S, and complete the entries of S to obtain flow conservation.

In this paper the feasibility question is studied in a more restrictive setting, where *all* the entries of the profile vector are considered. By formulating several necessary conditions, an upper bound is given on the number of feasible permutations, out of all permutations on q^ℓ elements. In addition, a linear programming technique is devised to decide the feasibility of a given permutation. Using this technique, an encoding algorithm for producing a large number of feasible permutations for any alphabet size q and any ℓ is given. Interestingly, some of the above results rely on an interpretation of the encoding process as a *Markov chain* on the DeBruijn graph. Further, it is also observed that if the entries of the profile vector are grouped according to equivalence classes of cyclic rotations, then any permutation is feasible. Even though this approach might seem simpler, it induces lower information rate and lower error resilience.

Finally, this problem may also be seen in a more general setting, beyond any applications for DNA storage. For example, it may be seen as a highly-restrictive variant of *constraint coding*, an area of coding theory that concerns the construction of strings with or without some prescribed substrings. This resemblance is apparent by observing that in our setting, *every* ℓ-substring is required to be more or less frequent than any other ℓ-substring.

II. Preliminaries

For an alphabet Σ of size q and a window size ℓ, let G_q^ℓ be the DeBruijn graph of order ℓ over Σ. That is, the node set $V(G_q^\ell)$ is Σ^ℓ, and for any two nodes u and v in $V(G_q^\ell)$, the edge set $E(G_q^\ell)$ contains (u,v) if the $(\ell-1)$-suffix of u equals the $(\ell-1)$-prefix of v. An edge (u,v) is labeled by a string $w \in \Sigma^{\ell+1}$ whose ℓ-prefix is u and whose ℓ-suffix is v.

For a string x over Σ, the ℓ-profile vector $p_x \in \mathbb{Z}^{q^\ell}$ (or simply, the profile vector) is a vector with q^ℓ non-negative integer entries $(p_x(w))_{w\in\Sigma^\ell}$, each of which contains the number of occurrences of the corresponding ℓ-substring in x, i.e., $p_x(w) = |\{i \mid x_i,\ldots,x_{i+\ell-1} = w\}|$ for all $w \in \Sigma^\ell$, where indices are taken modulo $|x|$. The entries of p_x may be

This work was supported in part by the Israel Science Foundation (ISF) under grant No. 130/14 and grant No. 1624/14.

978-1-5090-4096-4/17/$31.00 ©2017 IEEE

identified by either the set of nodes of G_q^ℓ or the set of edges of $G_q^{\ell-1}$, and the subscript x is omitted if clear from context.

The symmetric group S_{q^ℓ} is seen as the set of all bijective functions from Σ^ℓ to $\{0,1,\ldots,q^\ell-1\}$. For a permutation $\pi \in S_{q^\ell}$, a vector $p \in \mathbb{R}^{q^\ell}$ *satisfies* π if the relative order among the entries of p is identical to that of π, in which case we denote $p \vDash \pi$. Similarly, a string $x \in \Sigma^*$ (where Σ^* is the set of all closed strings) satisfies a permutation $\pi \in S_{q^\ell}$ if $p_x \vDash \pi$ (or $x \vDash \pi$ by abuse of notation), where p_x is the ℓ-profile vector of x. Further, a permutation π is *feasible* if there exists a string $x \in \Sigma^*$ such that $x \vDash \pi$. Clearly, only vectors with distinct entries can satisfy a permutation, and hence, not every string satisfies a permutation.

The main goals in this paper are to characterize, bound, and construct sets of feasible permutations in S_{q^ℓ}. To this end, given q and ℓ, let

$$\mathcal{F}_{q,\ell} \triangleq \{\pi \in S_{q^\ell} \mid \exists x \in \Sigma^*,\ x \vDash \pi\}, \text{ and}$$
$$R_{q,\ell} \triangleq \frac{\log_2 |\mathcal{F}_{q,\ell}|}{\log_2(q^\ell!)}.$$

It is evident that $\mathcal{F}_{q,1} = S_q$ for any q, and that $\mathcal{F}_{2,\ell} = \varnothing$ for every $\ell \geq 2$ (due to the structure of G_q^ℓ around the constant strings σ^ℓ). Therefore, the simplest set of parameters, that will be prominent in Section V, is $\ell = 2$ and $q = 3$.

A. Grouping by equivalence classes

Let $\sim$ be the equivalence relation of cyclic rotations on Σ^ℓ, i.e., $u \sim v$ if one can be obtained from the other by cyclic rotations, and let ${}^{\Sigma^\ell}/_{\sim}$ be the corresponding set of equivalence classes (that are often called *necklaces* [5]). It is known [5, p. 141, Eq. 4.63] that the size of any equivalence class in ${}^{\Sigma^\ell}/_{\sim}$ divides ℓ, and the size ψ_ℓ of ${}^{\Sigma^\ell}/_{\sim}$ is given by the following formula, in which ϕ is Euler's totient function.

$$\psi_\ell = \frac{1}{\ell}\sum_{d|\ell} \phi(d) q^{\ell/d}$$

For a profile vector $p \in \mathbb{Z}^{q^\ell}$, the vector $\tilde{p} \in \mathbb{Z}^{\psi_\ell}$ is an integer vector whose entries are identified by the elements of ${}^{\Sigma^\ell}/_{\sim}$. Each entry $\tilde{p}(C)$ for $C \in {}^{\Sigma^\ell}/_{\sim}$ contains the sum of entries in p that correspond to the elements of C.

Let $\tilde{S}_{\psi_\ell}$ be the set of bijective functions from ${}^{\Sigma^\ell}/_{\sim}$ to $\{0,1,\ldots,\psi_\ell-1\}$. A permutation $\pi \in \tilde{S}_{\psi_\ell}$ is feasible if there exists a string $x \in \Sigma^*$ such that $\tilde{p}_x \vDash \pi$. Since each equivalence class is a cycle in G_q^ℓ, and since G_q^ℓ is strongly connected, the following is easy to prove by choosing a proper path in G_q^ℓ.

Theorem 1. *For any q and any ℓ, all the permutations in $\tilde{S}_{\psi_\ell}$ are feasible.*

Theorem 1 suggests an alternative rank modulation scheme. However, grouping by equivalence classes diminishes the error resilience of the system. Further, it will be evident from Theorem 15 which follows, that a comparable rate can be achieved explicitly without grouping by equivalence classes.

B. Feasible permutations and Markov Chains

Some of the results which follow utilize an intriguing connection to Markov chains over DeBruijn graphs. This connection, which is described in this subsection, may be used for obtaining a randomized encoding algorithm.

A Markov chain [2] is an infinite series of random variables $X_0, X_1, \ldots$, whose states space is the vertex set of a graph (G_q^ℓ in this paper). For any natural n and for any set of vertices $i_0,\ldots,i_n$, and j, a Markov chain satisfies that

$$\Pr(X_{n+1} = j \mid X_n = i_n, X_{n-1} = i_{n-1}, \ldots, X_0 = i_0) = \Pr(X_{n+1} = j \mid X_n = i_n),$$

and the right hand side is independent of n. Clearly, since this process is memoryless, the transition probabilities can be described by a transition matrix M, such that $M_{i,j}$ is the probability that the random path traverses to vertex j, given that it is currently in vertex i. If v is a probability distribution on nodes at a certain step of the chain (i.e., v_i is the probability that the chain is currently in vertex i), then vM is the probability distribution in the following step.

A Markov chain is *irreducible* if for any states i and j, there exists an integer t such that the chain travels from state i to state j in t steps with a nonzero probability. A Markov chain is called *positive recurrent* if the mean return time to any state is finite. It is readily verified that any Markov chain on a DeBruijn graph with no zero transition probabilities is irreducible and positive recurrent, and we henceforth restrict our attention to such Markov chains.

It is a basic fact that a Markov chain on any finite graph has a unique *stationary distribution*, that is, a probability distribution p such that $pM = p$. Moreover, given *any* initial distribution v it is known that the chain tends to p as the number of steps tends to infinity, i.e., $vM^i \overset{i\to\infty}{\longrightarrow} p$. In the remainder of this section, $M \in \mathbb{R}^{q^\ell \times q^\ell}$ is a transition matrix of an irreducible and positive recurrent Markov chain on G_q^ℓ for some q and ℓ, and p is its stationary distribution. The next observation follows from the structure of the adjacency matrix of G_q^ℓ.

Observation 2. *For any $u \in \Sigma^{\ell-1}$, the stationary distribution p of M satisfies $\sum_{\sigma\in\Sigma} p(\sigma u) = \sum_{\sigma\in\Sigma} p(u\sigma)$.*

This lemma implies that a stationary distribution satisfies all *flow-conservation* constraints on the *edges* of $G_q^{\ell-1}$ [7]. Now, take a rational approximation of p, multiply it by the lcm of its entries' denominators, and place the resulting numbers on the edges of $G_q^{\ell-1}$. This implies, according to Observation 2, that there exists an Eulerian path in $G_q^{\ell-1}$ such that its corresponding ℓ-profile vector satisfies p, which gives rise to the following result.

Lemma 3. *For a given permutation $\pi \in S_{q^\ell}$, if there exists a transition matrix M whose stationary distribution satisfies π, then there exists a string x which satisfies π.*

The above discussion implies two possible algorithms for obtaining a string which satisfies a given stationary distribution p. A randomized one is given by defining the transition matrix

$$(M_p)_{a,b} = \begin{cases} \frac{p(v\sigma)}{\sum_{\tau\in\Sigma} p(v\tau)} & \text{if } (a,b) \text{ is an edge and } b = v\sigma, \\ 0 & \text{otherwise,} \end{cases}$$

whose stationary distribution is p, and following the resulting Markov chain for long enough. A deterministic one is given by finding an Eulerian path from the rational approximation of p, as described above.

In the sequel it is shown that these algorithms apply for any real vector χ that satisfies the flow conservation constraints (Lemma 10), and a method for deciding if such a vector exists and finding it for a given permutation is provided (Section IV). Further, the encoding algorithms in Section V operate by generating a large set of such vectors.

III. Upper Bound

The upper bound in this section relies on the following necessary condition for feasibility of a given permutation π. To formulate this condition, for a string $u \in \Sigma^{\ell-1}$ let $G_q^\ell(u)$ be the subgraph of G_q^ℓ which consists of the vertices $\{\sigma u\}_{\sigma\in\Sigma} \cup \{u\sigma\}_{\sigma\in\Sigma}$, that are connected as in G_q^ℓ. Given a permutation $\pi \in S_{q^\ell}$, color the edges of G_q^ℓ (and of $G_q^\ell(u)$ for all $u \in \Sigma^{\ell-1}$) in red and green such that an edge (a, b) is colored in green if $\pi(a) < \pi(b)$ and in red if $\pi(a) > \pi(b)$. The following constraint encapsulates the aforementioned flow conservation constraint.

Lemma 4. *If there exists $u \in \Sigma^{\ell-1}$ such that $G_q^\ell(u)$ contains an all-red perfect matching or an all-green perfect matching, then π is infeasible.*

A central tool in this section is *an arrangement*. For $A \subseteq \Sigma^\ell$, an arrangement of A is a vector of length $|A|$ that contains each element of A exactly once, and a *sub-arrangement* is obtained from an arrangement by deleting some of its entries. For a permutation π on a set $A \subseteq \Sigma^\ell$, i.e., an injective function from A to $\{0, \ldots, |A| - 1\}$, the corresponding *arrangement* is the vector $(\pi^{-1}(0), \ldots, \pi^{-1}(|A| - 1))$. The bound in this section is obtained by counting the number of infeasible arrangements on string sets of the form $V(G_q^\ell(u))$ for some u, and then expanding this bound to a large set of mutually disjoint subgraphs. The proof of the following lemma, which is omitted due to lack of space, uses the well-known Catalan numbers to enumerate the number of all-green and all-red perfect matchings in a subgraph $G_q^\ell(u)$.

Lemma 5. *For $u \in \Sigma^{\ell-1}$, the number of infeasible arrangements of $V(G_q^\ell(u))$ is at least $\frac{2}{q+1} \cdot (2q)!$.*

Mutually disjoint subgraphs $G_q^\ell(u)$ are obtained as follows.

Lemma 6. *If $\{u_i\}_{i=1}^k \subseteq \Sigma^{\ell-1}$ is an independent set of vertices in $G_q^{\ell-1}$, then the sets $V(G_q^\ell(u))$ are mutually disjoint.*

Let $\mathcal{U}$ be an independent set of size k in $G_q^{\ell-1}$ with no self loops, and let $\{\pi_i\}_{i\in[k]}$ be a set of arrangements on the respective nodes of $\{G_q^\ell(u)\}_{u\in\mathcal{U}}$. The following claim presents the number of arrangements on Σ^ℓ that contain π_i as a *sub-arrangement* for all $i \in [k]$. This claim is proven by considering the set of multi-permutations on the multi-set $\{1^{2q}, \ldots, k^{2q}\}$, and embedding π_i in the positions which contain i for each $i \in [k]$.

Lemma 7. *There exist $\frac{q^\ell!}{(2q)!^k}$ arrangements π on Σ^ℓ that contain π_i on $V(G_q^\ell(u_i))$ for any $i \in [k]$.*

Fortunately, the maximum value of k was studied in [10], where it is called the *loopless independence number* $\alpha^*(q, \ell)$.

Lemma 8. *[10] $\alpha^*(q, \ell) \geq \frac{q^\ell - q^{\ell-2}}{3}$.*

Using Lemma 5, Lemma 7, and Lemma 8, the number of arrangements that contain an infeasible sub-arrangement on *some* subgraph $G_q^\ell(u)$ can be bounded. Unfortunately, the resulting bound does not seem to provide a non-trivial upper bound on the rate $R_{q,\ell}$.

Theorem 9. *For any q and any ℓ, $|\mathcal{F}_{q,\ell}| \leq \left(\frac{q-1}{q+1}\right)^{\frac{q^\ell - q^{\ell-2}}{3}} \cdot q^\ell!$.*

IV. A polynomial algorithm for deciding feasibility

In this section it is shown that a given permutation $\pi \in S_{q^\ell}$ can be decided to be feasible or not in time polynomial in q^ℓ. If it is feasible, the time complexity of producing a string which satisfies it depends on its minimal length, a topic yet to be studied. The given algorithm relies on the following simple claim, which resembles Observation 2 and its subsequent discussion.

Lemma 10. *A permutation $\pi \in S_{q^\ell}$ is feasible if and only if there exists a vector $\chi \in \mathbb{R}^{q^\ell}$ such that $\pi \vDash \chi$ and for all $u \in \Sigma^{\ell-1}$, $\sum_{\sigma\in\Sigma} \chi(u\sigma) = \sum_{\sigma\in\Sigma} \chi(\sigma u)$.*

Given a permutation π, Lemma 10 gives rise to the linear programming problem below for deciding its feasibility. In what follows, for a set $A \subseteq \Sigma^\ell$ the notation $\mathbf{1}A$ stands for the binary characteristic vector of A, and '$\cdot$' denotes the ordinary inner product.

Variables $\{\chi_v \mid v \in \Sigma^\ell\}$.
Objective None.
Constraints

- $\chi(u) - \chi(v) > 0$ for all distinct u and v in Σ^ℓ such that $\pi(u) > \pi(v)$.
- $\chi \cdot (\mathbf{1}\{v\sigma \mid \sigma \in \Sigma\} - \mathbf{1}\{\sigma v \mid \sigma \in \Sigma\}) = 0$ for all $v \in \Sigma^{\ell-1}$.

According to Lemma 10, determining the feasibility of this system is equivalent to determining the feasibility of the given permutation π. Since deciding the feasibility of a linear programming problem can be done in polynomial time, so does the feasibility of a given permutation.

Given the vector χ, the corresponding permutation may be found by sorting its entries. The corresponding string may be generated by shifting the entries of χ by an additive constant to obtain a positive vector, normalizing the result by the sum of entries, and following either the randomized or the deterministic algorithms that were mentioned after Lemma 3. For this reason, the encoding algorithms in the sequel focus on finding the vector χ, rather than finding the permutation itself.

V. Encoding Algorithms

In this section encoding algorithms are given for any ℓ and any q. These algorithms provide a lower bound on the size of $\mathcal{F}_{q,\ell}$ for the respective parameters. Since an additive structure of the alphabet is required, it is assumed in this section that $\Sigma = \mathbb{Z}_q$. From the reasons that were discussed at the end of Section IV, the algorithms that are detailed in this section focus on providing the vector χ (Lemma 10). In Subsection V-A an algorithm for $\ell = 2$ and any q is given; and in Subsection V-B it is used as the base case for another algorithm which obtains feasible permutations for any ℓ and any q.

To clearly describe the constraints on the vector χ, by abuse of notation it is considered either as a vector in $\mathbb{R}^{q^\ell}$ or as a *matrix* in $\mathbb{R}^{q^{\ell-1}\times q^{\ell-1}}$. That is, given a string u in $\mathbb{Z}_q^\ell$, the notation $\chi(u)$ stands for the u-th entry of χ when seen as a vector, and given two strings w, w' in $\Sigma^{\ell-1}$, the notation $\chi(w, w')$ denotes the (w, w') entry when seen as a matrix, where

$$\chi(w, w') \triangleq \begin{cases} \chi(u) & \text{if } (w, w') \text{ is a } u\text{-labelled edge in } E(G_q^\ell) \\ 0 & \text{else} \end{cases}.$$

Algorithm 1: $A_q(m)$, an encoding algorithm for $\ell = 2$ and any q.

Data: A repository R of all $f_{3,2}$ feasible matrices for $q = 3$ and $\ell = 2$.
Input : An information vector $m \in I_q$.
Output: A feasible matrix $\chi \in \mathbb{R}^{q\times q}$ (which represents a feasible vector $\chi \in \mathbb{R}^{q^2}$).
if $q = 3$ **then** return the m-th feasible matrix in R;
Denote $m = (m', \pi_q)$ for $m' \in I_{q-1}$ and $\pi_q \in S_q$.
Apply $A_{q-1}(m')$ to get a matrix χ', and for $i, j \in \mathbb{Z}_{q-1}$ denote $(\chi')_{i,j} = x_{i,j}$.
Choose ε such that

$$\varepsilon < \frac{\min\{|x_{i,j} - x_{s,t}| : i, j, s, t \in \mathbb{Z}_{q-1} \text{ and } (i,j) \neq (s,t)\}}{q}.$$

Choose[1] $y \triangleq (y_0, \ldots, y_{q-1}) \in \mathbb{R}^q$ such that $y \vDash \pi_q$, and such that the following matrix contains distinct positive entries

$$\chi \triangleq \begin{pmatrix} y_0 & y_1 - \frac{(q-1)(q-2)}{2} \cdot \varepsilon & y_2 & \cdots & y_{q-1} \\ y_1 & x_{0,0} & x_{0,1} & \cdots & x_{0,q-2} \\ y_2 - \varepsilon & x_{1,0} + \varepsilon & x_{1,1} & \cdots & x_{1,q-2} \\ y_3 - 2\varepsilon & x_{2,0} + 2\varepsilon & x_{2,1} & \cdots & x_{2,q-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y_{q-1} - (q-2)\varepsilon & x_{q-2,0} + (q-2)\varepsilon & x_{q-2,1} & \cdots & x_{q-2,q-2} \end{pmatrix}.$$

return χ.

Using this notation, for $v \in \mathbb{Z}_q^{\ell-1}$ the constraint $\sum_{\sigma\in\mathbb{Z}_q} \chi(v\sigma) = \sum_{\sigma\in\mathbb{Z}_q} \chi(\sigma v)$ may be written as $\sum_{u\in\mathbb{Z}_q^{\ell-1}} \chi(v,u) = \sum_{u\in\mathbb{Z}_q^{\ell-1}} \chi(u,v)$, i.e., the v-th row sum equals the v-th column sum. A vector (matrix) χ which satisfies these constraints and satisfies a permutation is called a *feasible vector (matrix)*.

A. An encoding algorithm for $\ell = 2$ and any q

This algorithm operates recursively on q, where the base case is $q = 3$. To resolve the base case, the set $\mathcal{F}_{3,2}$ (or the corresponding set of feasible matrices) should be kept in a repository. It can be shown both combinatorially and computationally that the size of $\mathcal{F}_{3,2}$ is $f_{3,2} \triangleq 30240$. In what follows, the discussion refers to Algorithm 1, in which the information vector is taken from the set $I_q \triangleq [f_{3,2}] \times [4!] \times [5!] \times \cdots \times [q!]$.

By the choice of ε and the choice of y, it is evident that the output χ contains distinct and positive entries. To prove the correctness of the algorithm, it ought to be shown that the row and column sum constraint from Lemma 10 is satisfied, and that different information vectors result in different permutations. First, by summing across the rows and columns of the matrix χ in Algorithm 1, the following holds.

Lemma 11. *For all* $\tau \in \mathbb{Z}_q$, $\sum_{\sigma\in\mathbb{Z}_q} \chi(\tau,\sigma) = \sum_{\sigma\in\mathbb{Z}_q} \chi(\sigma,\tau)$.

Second, the fact that Algorithm 1 is injective is as follows.

Lemma 12. *If u and v are distinct information vectors in I_q such that $A_q(u) = \chi_u, A_q(v) = \chi_v$ and $\chi_u \vDash \pi_u, \chi_v \vDash \pi_u$ for some permutations π_u and π_v, then $\pi_u \neq \pi_v$.*

Proof. Note that according to the choice of ε, the relative order among the entries of χ' is preserved. Hence, if $u_i \neq v_i$ for some entry i, then the respective permutations between the y_j-s in stage i of the algorithm (or the chosen permutation from R if $i = 1$) are distinct. Hence, the permutation on the corresponding entries of χ_u and χ_v are also distinct, and the claim follows. □

Corollary 13. *For any q,*

$$|\mathcal{F}_{q,2}| \geq f_{3,2} \cdot \prod_{i=0}^{q-4}(q-i)! \quad \text{and} \quad \lim_{q\to\infty} R_{q,2} \geq \frac{1}{4}.$$

B. An encoding algorithm for any ℓ and any q

In this section the recursive algorithm from Subsection V-A is used to obtain an encoding algorithm for any ℓ and any q. Inspired by [9] and [11], the recursive step of the algorithm relies on embedding a feasible matrix $\chi_{\ell-1}$ in *homomorphic pre-images* of $G_q^{\ell-1}$ in G_q^ℓ, and breaking ties that emerge. The information vector at hand is encoded *into* the particular way that these ties are broken.

Definition 14. *[6] For graphs G and H, a function $f : V(G) \to V(H)$ is a (graph) homomorphism if for each pair of vertices u, v in $V(G)$, if (u, v) is an edge in G then $(f(u), f(v))$ is an edge in H.*

The algorithm which follows relies on the following homomorphism, and yet, many other homomorphisms exist, and either of them may be used similarly.

$$D_\ell : \mathbb{Z}_q^\ell \to \mathbb{Z}_q^{\ell-1}$$
$$D_\ell(v) \triangleq (v_0 + v_1, v_1 + v_2, \ldots, v_{\ell-2} + v_{\ell-1}), \text{ and}$$
$$D : \mathbb{Z}_q^* \to \mathbb{Z}_q^*$$
$$D(v) \triangleq D_{|v|}(v).$$

It is an easy exercise to prove that for any ℓ, the function D is a q to 1 surjective homomorphism from G_q^ℓ to $G_q^{\ell-1}$. That is, for every $u \in \mathbb{Z}_q^{\ell-1}$, the set $D^{-1}(u)$ contains exactly q elements. Moreover, it is readily verified that this set may be written as

$$D^{-1}(u) = \{v_0, \ldots, v_{q-1}\},$$

where $v_{i,0} = i$ for all $i \in \mathbb{Z}_q$. Similarly, for every $u \in \mathbb{Z}_q^{\ell-1}$,

$$\{D(\sigma u)\}_{\sigma\in\mathbb{Z}_q} = \{\sigma D(u)\}_{\sigma\in\mathbb{Z}_q}, \text{ and}$$
$$\{D(u\sigma)\}_{\sigma\in\mathbb{Z}_q} = \{D(u)\sigma\}_{\sigma\in\mathbb{Z}_q}.$$

[1] Since y is a real vector, it is clear that such a vector exists and may easily be found.

Algorithm 2: $B_q^\ell(m)$, an encoding algorithm for any ℓ and any q.

Input : An information vector $m \in J_q^\ell$.
Output: A feasible matrix $\chi \in \mathbb{R}^{q^{\ell-1} \times q^{\ell-1}}$ (which represents a feasible vector $\chi \in \mathbb{R}^{q^\ell}$.)
if $\ell = 2$ **then** return $A_q(m)$ (Algorithm 1);
Denote $m = (m', P)$, where $P \in \mathcal{P}_\ell$ and $m' \in J_q^{\ell-1}$.
Apply $B_q^{\ell-1}(m')$ to obtain a matrix $\chi_{\ell-1}$.
Multiply $\chi_{\ell-1}$ by a positive constant[2] such that the absolute differences between any two entries in the resulting matrix are at least 2.
foreach $v \in \mathbb{Z}_q^\ell$ **do**
  Denote $v = v_0 v'$, where $v_0 \in \mathbb{Z}_q$ and $v' \in \mathbb{Z}_q^{\ell-1}$.
  Denote $D(v) \triangleq w = (w_0 w' w_{\ell-2})$, where $w_0, w_{\ell-2} \in \mathbb{Z}_q$ and $w' \in \mathbb{Z}_q^{\ell-3}$.
  Define

$$\chi_\ell(v) = \begin{cases} \chi_{\ell-1}(D(v)) + \frac{P(D(v))(v_0)}{q^{\langle w_0, w_{\ell-2}\rangle}} & \text{if } D(v) \in \mathcal{A}_\ell \\ \chi_{\ell-1}(D(v)) - \sum_{\mu \neq v_0} \frac{P(\mu - v_0, w' w_{\ell-2})(\mu)}{q^{\langle \mu - v_0, w_{\ell-2}\rangle}} & \text{if } w_0 = 0, w_{\ell-2} \neq 0 \\ \chi_{\ell-1}(D(v)) - \sum_{\tau \neq 0} \frac{P(w_0 w', \tau)(v_0)}{q^{\langle w_0, \tau\rangle}} & \text{if } w_0 \neq 0, w_{\ell-2} = 0 \\ \chi_{\ell-1}(D(v)) + \sum_{\mu \neq v_0} \sum_{\tau \neq 0} \frac{P(\mu - v_0, w', \tau)(\mu)}{q^{\langle \mu - v_0, \tau\rangle}} & \text{if } w_0 = w_{\ell-2} = 0 \end{cases}$$

end
return χ_ℓ.

The information vector is taken from the set J_q^ℓ, defined as follows.

$J_q^\ell \triangleq I_q \times \mathcal{P}_3 \times \mathcal{P}_4 \times \cdots \times \mathcal{P}_\ell$, where
$\mathcal{P}_i \triangleq \{P \mid P : \mathcal{A}_i \to S_q\}$, and
$\mathcal{A}_i \triangleq \{u \in \mathbb{Z}_q^{i-1} \mid 0 \notin \{u_0, u_{i-2}\}\}$ for all $i \in \{3, \ldots, \ell\}$.

That is, the information vector is of the form $m = (m', P_3, \ldots, P_\ell)$, where P_i is a function from $\mathcal{A}_i$ to S_q, and $m' \in I_q$ (see Subsection V-A). In addition, for i and j in $\mathbb{Z}_q$, let $\langle i, j\rangle$ be $\langle i, j\rangle \triangleq i + j \cdot q + 1$.

In stage ℓ, Algorithm 2 relies on embedding the matrix $\chi_{\ell-1}$, which results from stage $\ell - 1$, in entries of χ_ℓ that correspond to homomorphic pre-images of $G_q^{\ell-1}$ in G_q^ℓ. Since the homomorphism D is q to 1, this results in $q^{\ell-1}$ sets of entries in χ_ℓ, of q elements each, that contain identical entries. These equalities are broken by adding small constants to each set, where these constants are ordered according to the permutations in S_q that appear in the current entry of the information vector. To maintain the row and column sum constraint (Lemma 10), a unique entry in every row and every column is chosen, and its addition is adjusted to cancel out the sum of additions from its respective row or column.

To verify the correctness of Algorithm 2, it must be shown that the row and column sum constraint (Lemma 10) is satisfied, that the entries are distinct, and that it is injective. Note that the additions to $v \in \mathbb{Z}_q^\ell$ such that $D(v) \notin \mathcal{A}_\ell$ are chosen to cancel out the additions in their respective rows and columns, and hence the row and column sums of χ_ℓ are identical to those of $\chi_{\ell-1}$. To prove that the entries are distinct, it is readily observed that all ties in the homomorphic pre-images of $G_q^{\ell-1}$ in G_q^ℓ are broken with fractions that have distinct q-ary expansions. The fact that Algorithm 2 is injective follows similarly to Lemma 12. Full proofs of these facts will appear in future version of this paper. In addition, Algorithm 2 implies the following.

Theorem 15. *For any q and ℓ,*

$$|\mathcal{F}_{q,\ell}| \geq f_{3,2} \cdot \left(\prod_{i=0}^{q-4} (q-i)! \right) \cdot \left(\prod_{i=3}^{\ell} (q!)^{q^{i-1} - 2q^{i-2} + q^{i-3}} \right),$$

and hence, $\lim_{q \to \infty} R_{q,\ell} \geq \frac{1}{\ell}$ for any fixed $\ell \geq 3$.

VI. Discussion

In this paper a rank modulation scheme for DNA storage was studied. In particular, an upper bound for the number of feasible permutations was given by formulating a necessary condition, and an encoding algorithm was devised.

Other than improving the results in this paper, for future research we would also like to find the minimum n for which all the permutations in $\mathcal{F}_{q,\ell}$ have a corresponding string of length n. Furthermore, we would like to endow $\mathcal{F}_{q,\ell}$ with a proper metric, such as Kendall's τ, and find good codes by this metric.

References

[1] J. Bornholt, R. Lopez, D. M. Carmean, L. Ceze, G. Seelig, and K. Strauss, "A DNA-based archival storage system," *In Proc. ASPLOS*, pp. 637–649, 2016.
[2] P. Brémaud. Markov chains: Gibbs fields, Monte Carlo simulation, and queues, vol. 31, Springer Science & Business Media, 2013.
[3] G. M. Church, Y. Gao, S. Kosuri, "Next-generation digital information storage in DNA," *Science*, 337.6102, pp. 1628-1628, 2012.
[4] N. Goldman, P. Bertone, S. Chen, C. Dessimoz, E. M. LeProust, B. Sipos, and E. Birney, "Towards practical, high-capacity, low-maintenance information storage in synthesized DNA," *Nature*, vol. 494, no. 7435, pp. 77-80, 2013.
[5] R. L. Graham, D. L. Knuth, and O. Patashnik. Concrete Mathematics: A Foundation for Computer Science (2nd Edition). Addison-Wesley, 1994.
[6] J. L. Gross and J. Yellen. Handbook of graph theory. CRC press, 2004.
[7] P. Jacquet, C. Knessl and W. Szpankowski, "Counting Markov Types, Balanced Matrices, and Eulerian Graphs," *IEEE Transactions on Information Theory*, vol. 58, no. 7, pp. 4261–4272, 2012.
[8] H. M. Kiah, J. G. Puleo, and O. Milenkovic, "Codes for DNA sequence profiles," *IEEE Transactions on Information Theory*, vol. 62, no. 6, pp. 3125–3146, 2016.
[9] A. Lempel, "On a homomorphism of the de Bruijn graph and its applications to the design of feedback shift registers," *IEEE Transactions on Computers*, vol. 100, no. 12, pp. 1204–1209, 1970.
[10] N. Lichiardopol, "Independence number of de Bruijn graphs," *Discrete mathematics*, vol. 306, no. 12, pp. 1145–1160, 2006.
[11] M. Liu, "Homomorphisms and automorphisms of 2-D de Bruijn-Good graphs," *Discrete Mathematics*, vol. 85, no. 1, pp. 105–109, 1990.
[12] A. S. Motahari, G. Bresler, and N. C. David, "Information theory of DNA shotgun sequencing," *IEEE Transactions on Information Theory*, vol. 59, no .10, pp. 6273–6289, 2013.

[2]Notice that multiplying a feasible matrix by any positive constant does not compromise the row and column sum constraints, and does not change the relative order of the entries.