IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 15 December 2023, accepted 28 February 2024, date of publication 11 March 2024, date of current version 19 March 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3376476

RESEARCH ARTICLE

Quasi-Enumerative Coding of Balanced Run-Length Limited Codes

FILIP PALUNČIĆ, (Member, IEEE), AND B. T. MAHARAJ, (Senior Member, IEEE)
Department of Electrical, Electronic and Computer Engineering, University of Pretoria, Pretoria 0002, South Africa

Corresponding author: Filip Palunčić (filip.paluncic@up.ac.za)

This work was supported by the SENTECH Chair in Broadband Wireless Multimedia Communication.

ABSTRACT Various methods have been proposed for the construction of balanced run-length limited codes. Amongst these methods is the enumerative coding approach by Kurmaev. The advantage of this approach is that the code has maximum cardinality and thus approaches capacity with increasing codeword length. However, enumerative coding has the disadvantage of becoming prohibitively complex for large codeword lengths. We propose an alternative enumerative coding method that reduces the encoding and decoding complexity. We call it quasi-enumerative coding as it does not follow a strict lexicographic order, but retains a one-to-one mapping between rank and the corresponding balanced run-length limited codeword.

INDEX TERMS Enumerative coding, balanced codes, run-length limited codes.

I. INTRODUCTION

Constrained codes are an important class of codes that impose beneficial constraints on the encoded data with various applications in magnetic and optical storage media [1], [2], [3]. Examples of such codes include run-length limited (RLL) codes, which limit the number of consecutive zeros between ones, and balanced codes, where the number of zeros equals the numbers of ones in each codeword. In the case of RLL codes, the minimum run-length constraint limits inter-symbol interference, while the maximum run-length constraint ensures sufficient transitions for clock recovery. Balanced codes are an instance of spectral-shaping codes where the low frequency region of the spectrum of the encoded data is suppressed or reduced.

Balanced RLL codes consist of codewords which are both balanced and run-length limited simultaneously. Examples of such codes in literature include the codes by Immink et al. [4] and Palunčić et al. [5] which are based on adaptions of the Knuth's balancing procedure for random (non-RLL) binary sequences [6] extended to RLL sequences, and the enumerative coding approach by Kurmaev [7]. While the codes based on Knuth-like balancing of RLL sequences are well suited to long codeword lengths due to the simplicity of the balancing method, they are not of maximum cardinality and also require the addition of a suffix to identify the balancing point. The advantage of enumerative coding by Kurmaev is that the codes include all possible balanced RLL codewords resulting in maximum codebook cardinality. However, it has the disadvantage of being prohibitively complex in terms of computation and/or storage requirements for larger codeword lengths.

Manada and Morita [8] prove that the capacity of balanced RLL codes is equal to that of the corresponding RLL codes. Hence, the addition of the balancing constraint to RLL sequences does not alter the resultant capacity. Furthermore, Palunčić and Maharaj [9] use bivariate generating functions to count the exact number of balanced RLL sequences of a particular length.

In this paper we present various enumerative coding constructions which reduce the encoding and decoding complexity as compared to the enumerative coding technique of Kurmaev. The first proposed construction has maximum cardinality assuming that codeword boundaries coincide with run-length boundaries. The second proposed construction is an adaption of enumerative coding of permutation RLL codes developed by Milenkovic and Vasić [10]. The third construction is a simplified version of the previous construction. The cardinality, i.e., the code rate, of the constructions decreases from the first to the third; however, the encoding and decoding

The associate editor coordinating the review of this manuscript and approving it for publication was Khmaies Ouahada.

complexity decreases from first to the third. Therefore, there is a trade-off between code rate and complexity between the three constructions.

Traditional enumerative coding adheres to a strict lexicographic ordering of the codewords. The novelty of the approach proposed in this paper, which we term *quasi-enumerative coding*, relaxes the strict lexicographic ordering but retains a one-to-one mapping between index and codeword. This relaxation allows for the construction of more efficient codes in comparison with existing enumerative codes in literature.

This paper is organized as follows. Preliminary concepts and notation are introduced in Section II, while results pertinent to balanced RLL codes are summarized in Section III. The quasi-enumerative coding constructions for balanced RLL codes are presented in Section IV. Section V presents a performance comparison in terms of complexity and code rate efficiency and the paper's concluding remarks are contained in Section VI.

II. PRELIMINARIES AND NOTATION

Let $\boldsymbol{x} = (x_1 x_2 \dots x_{2m}) \in \{0,1\}^{2m}$ be a (d,k)-constrained word. A word is said to be (d,k)-constrained if two consecutive 1s are separated by at least d and at most k 0s. Assume that a (d,k)-constrained word $\boldsymbol{x}$ is obtained through the concatenation of the valid segments

$$1\underbrace{00\dots0}_{d}, 1\underbrace{000\dots0}_{d+1}, \dots, 1\underbrace{0000\dots0}_{k},$$

where the lengths of these segments are in the set $\{d+1, d+2, \dots, k+1\}$. These segments are referred to as *run-length segments*. Hence, it is assumed that $x_1 = 1$ which means that (d,k)-constrained words $\boldsymbol{x}$ can be concatenated without violating the (d,k) constraints. A (d,k)-constrained word $\boldsymbol{x}$ can also be represented as a $(d+1, k+1)$-RLL word $\boldsymbol{z} = (z_1 z_2 \dots z_{2m}) \in \{-1,+1\}^{2m}$ via a simple bijective mapping [4]

$$\begin{aligned}\hat{z}_i &= \hat{z}_{i-1} \oplus x_i,\\ z_i &= 2\hat{z}_i - 1,\end{aligned}$$

and

$$x_i = (-z_{i-1} z_i + 1)/2,$$

where $1 \le i \le 2m$, $\hat{z}_0 \triangleq 1$ and $\oplus$ denotes modulo 2 addition. Clearly, $x_i = 1$ corresponds to a transition from -1 to $+1$ or vice versa from z_{i-1} to z_i, while $x_i = 0$ corresponds to no transition. It should be noted that in the concatenation of (d,k)-constrained or $(d+1, k+1)$-RLL words, the definition of $\hat{z}_0$ is only required for the first word, where for subsequent words the role of $\hat{z}_0$ is taken over by the last bit of the previous word.

Let $w(\boldsymbol{x})$ denote the weight of $\boldsymbol{x}$, i.e.,

$$w(\boldsymbol{x}) \triangleq \sum_{i=1}^{2m} x_i.$$

Clearly, $w(\boldsymbol{x})$ equals the number of run-length segments in $\boldsymbol{x}$ and the number of runs in $\boldsymbol{z}$. The *digital sum* or *charge* of $\boldsymbol{z}$ is defined as

$$\sigma(\boldsymbol{z}) \triangleq \sum_{i=1}^{2m} z_i.$$

A (d,k)-constrained word $\boldsymbol{x}$ is said to be *balanced* if $\sigma(\boldsymbol{z}) = 0$, i.e., if $\boldsymbol{z}$ has an equal number of -1s and $+1$s.

A (d,k)-constrained word or its equivalent $(d+1, k+1)$-RLL word can be uniquely represented in terms of the run-length segment or run lengths. Consider a (d,k)-constrained word $\boldsymbol{x}$ consisting of $w(\boldsymbol{x})$ run-length segments. Then $\boldsymbol{\rho''}(\boldsymbol{x}) = (\rho_1'' \rho_2'' \dots \rho_w'') \in \{d+1, d+2, \dots, k+1\}^w$ is the *run-length representation* of $\boldsymbol{x}$ where ρ_i'', $1 \le i \le w$, is the length of the ith run-length segment in $\boldsymbol{x}$. Equivalently, ρ_i'' is the length of the ith run in $\boldsymbol{z}$. For our purposes, a more amenable run-length representation is $\boldsymbol{\rho}(\boldsymbol{x}) = (\rho_1 \rho_2 \dots \rho_w) \triangleq \boldsymbol{\rho''}(\boldsymbol{x}) - [d+1]^w \in \{0, 1, \dots, r-1\}^w$, where $r \triangleq k - d + 1$ and the notation $[a]^w$ denotes a word consisting of w symbols a. It is clear that for a given run-length representation $\boldsymbol{\rho}$ of $\boldsymbol{x}$, the length of $\boldsymbol{x}$ is

$$w(d+1) + \sum_{i=1}^{w} \rho_i.$$

For convenience, in the remainder of the paper, the argument $\boldsymbol{x}$ will be dropped and $w(\boldsymbol{x})$ and $\boldsymbol{\rho}(\boldsymbol{x})$ will only be denoted as w and $\boldsymbol{\rho}$, respectively.

The notation $\langle ., . \rangle$ is used to denote the interleaving/deinterleaving operation. More specifically, if $\boldsymbol{\rho}^{(1)} = (\rho_1^{(1)} \rho_2^{(1)} \dots \rho_{w^{(1)}}^{(1)})$ and $\boldsymbol{\rho}^{(2)} = (\rho_1^{(2)} \rho_2^{(2)} \dots \rho_{w^{(2)}}^{(2)})$, then the *interleaving* operation, which produces the interleaved word $\boldsymbol{\rho}$, is defined as

$$\begin{aligned}\boldsymbol{\rho} &= (\rho_1 \rho_2 \rho_3 \rho_4 \dots \rho_{w-1} \rho_w)\\ &\equiv \langle \boldsymbol{\rho}^{(1)}, \boldsymbol{\rho}^{(2)} \rangle\\ &= (\rho_1^{(1)} \rho_1^{(2)} \rho_2^{(1)} \rho_2^{(2)} \dots \rho_{w^{(1)}}^{(1)} \rho_{w^{(2)}}^{(2)})\end{aligned}$$

if $w^{(1)} = w^{(2)}$ or

$$\begin{aligned}\boldsymbol{\rho} &= (\rho_1 \rho_2 \rho_3 \rho_4 \dots \rho_{w-1} \rho_w)\\ &\equiv \langle \boldsymbol{\rho}^{(1)}, \boldsymbol{\rho}^{(2)} \rangle\\ &= (\rho_1^{(1)} \rho_1^{(2)} \rho_2^{(1)} \rho_2^{(2)} \dots \rho_{w^{(1)}-1}^{(1)} \rho_{w^{(2)}}^{(2)} \rho_{w^{(1)}}^{(1)})\end{aligned}$$

if $w^{(1)} = w^{(2)} + 1$, where $w = w^{(1)} + w^{(2)}$. Therefore, all the odd indices in $\boldsymbol{\rho}$ are composed from $\boldsymbol{\rho}^{(1)}$ and all the even indices in $\boldsymbol{\rho}$ are composed from $\boldsymbol{\rho}^{(2)}$. Conversely, the *deinterleaving* operation produces the subwords $\boldsymbol{\rho}^{(1)}$ and $\boldsymbol{\rho}^{(2)}$ from $\boldsymbol{\rho} = (\rho_1 \rho_2 \dots \rho_w)$ and is denoted as $\langle \boldsymbol{\rho}^{(1)}, \boldsymbol{\rho}^{(2)} \rangle \equiv \boldsymbol{\rho}$.

Example 1: Consider the $(1,3)$-constrained word

$$\boldsymbol{x} = (101000100100100010100100).$$

Then

$$\begin{aligned}\boldsymbol{z} = (&- - + + + + - - - + + + - -\\ &- - + + - - - + + +),\end{aligned}$$

where '+' denotes '+1' and '−' denotes '−1'. Clearly, $\boldsymbol{z}$ is balanced as it contains an equal number of '+'s and '−'s. The run-length representation is $\boldsymbol{\rho} = (02112011) \equiv \langle(0121), (2101)\rangle$. □

III. BALANCED RLL CODES

Restricted integer compositions play a fundamental role in counting the number of balanced RLL words and also form the basis of one of the quasi-enumerative coding schemes presented in the next section. Hence, restricted integer compositions are formally defined.

Definition 1: A restricted composition of an integer n is a sequence $(\rho'_1\rho'_2\dots\rho'_w)$ of integers, where $\rho'_i \in \{1, 2, \dots, r\}$, $1 \le i \le w$, such that

$$\rho'_1 + \rho'_2 + \ldots + \rho'_w = n.$$

Let $R(n, w; r)$ be the number of such compositions. It is known that [11, Eq. (42)], [12, p. 441]

$$R(n, w; r) = \sum_{j=0}^{w} (-1)^j \binom{w}{j} \binom{n - jr - 1}{w - 1}. \tag{1}$$

If we consider a (d, k)-constrained word $\boldsymbol{x}$ consisting of w run-length segments, by removing d zeros from each run-length segment, the equivalent of a restricted integer composition is obtained. If $\boldsymbol{\rho'} \triangleq (\rho'_1\rho'_2\dots\rho'_w)$, then $\boldsymbol{\rho'} = \boldsymbol{\rho} + [1]^w = \boldsymbol{\rho''} - [d]^w$. It is easy to see that the number of (d, k)-constrained words $\boldsymbol{x}$ of length m consisting of w run-length segments is $R(m - wd, w; k - d + 1)$. We note that the expression for $R(n, w; r)$ from (1) can also be obtained from the formula for $A_n^v(d, k, r)$ [7, Eq. (9)], the number of (d, k)-constrained words of length n, weight v, and at most r trailing zeros, by setting $r = 0$ (removing the trailing run of zeros), $w = v - 1$ (removing the trailing run-length segment), $m = n - 1$ (removing the 1 of the trailing run-length segment), and $r = q = k - d + 1$.[1]

In [9], bivariate generating functions are derived which count the number of balanced RLL words of a particular length. The derivation is based on the observation that any balanced RLL word can be obtained by interleaving two RLL words of equal length and consisting of:

1) the same number of run-length segments, or
2) differing by one in the number of run-length segments.

More precisely, any balanced (d, k)-constrained, or $(d + 1, k + 1)$-RLL, word $\boldsymbol{\rho}$ of length w can be obtained by interleaving $\boldsymbol{\rho}^{(1)}$ of length $w^{(1)}$ and $\boldsymbol{\rho}^{(2)}$ of length $w^{(2)}$, i.e., $\boldsymbol{\rho} \equiv \langle\boldsymbol{\rho}^{(1)}, \boldsymbol{\rho}^{(2)}\rangle$, where $w = w^{(1)} + w^{(2)}$ and

$$\sum_{i=1}^{w^{(1)}} \rho_i^{(1)} = \sum_{i=1}^{w^{(2)}} \rho_i^{(2)}$$

when $w^{(1)} = w^{(2)}$, and

$$d + 1 + \sum_{i=1}^{w^{(1)}} \rho_i^{(1)} = \sum_{i=1}^{w^{(2)}} \rho_i^{(2)}$$

when $w^{(1)} = w^{(2)} + 1$.

Let $B(2m, d, k)$ be the number of balanced (d, k)-constrained words of length $2m$. Noting that a (d, k)-constrained word of length m can consist of at least $w_{\min} = \lceil \frac{m}{k+1} \rceil$ and at most $w_{\max} = \lfloor \frac{m}{d+1} \rfloor$ run-length segments, it follows that

$$\begin{aligned} &B(2m, d, k) \\ &= R^2(m - w_{\min}d, w_{\min}; k - d + 1) \\ &\quad + \sum_{i=w_{\min}+1}^{w_{\max}} \Big(R(m - (i-1)d, i - 1; k - d + 1) \\ &\quad \times R(m - id, i; k - d + 1) \\ &\quad + R^2(m - id, i; k - d + 1) \Big). \end{aligned}$$

Example 2: Consider balanced $(1, 3)$-constrained words of length 20 ($m = 10$). Then, the weights of the interleaved subwords can be

$$\left\lceil \frac{m}{k+1} \right\rceil \le w \le \left\lfloor \frac{m}{d+1} \right\rfloor$$
$$3 \le w \le 5.$$

For $w = 3$:

$$\begin{aligned} R(n, w; r) &= R(m - wd, w; r) \\ &= R(7, 3; 3) \\ &= \sum_{j=0}^{3} (-1)^j \binom{3}{j} \binom{7 - j3 - 1}{2} \\ &= \binom{3}{0}\binom{6}{2} - \binom{3}{1}\binom{3}{2} \\ &= 15 - 9 = 6, \end{aligned}$$

and for $w = 4$:

$$\begin{aligned} R(n, w; r) &= R(m - wd, w; r) \\ &= R(6, 4; 3) \\ &= \sum_{j=0}^{4} (-1)^j \binom{4}{j} \binom{6 - j3 - 1}{3} \\ &= \binom{4}{0}\binom{5}{3} \\ &= 10, \end{aligned}$$

and for $w = 5$:

$$\begin{aligned} R(n, w; r) &= R(m - wd, w; r) \\ &= R(5, 5; 3) \end{aligned}$$

[1] Note that the variables n, v and q follow the notation from Kurmaev's paper, which are equivalent to m, w and r here. The r (number of trailing zeroes) from Kurmaev's paper should not be confused with r in this paper (alphabet size for the run-length representation).

TABLE 1. List of all (1, 3)-constrained words of length $m = 10$ and weight $w \in \{3, 4, 5\}$.

	$w=3$	$w=4$	$w=5$
0	(022)	(0002)	(00000)
1	(112)	(0011)	
2	(121)	(0020)	
3	(202)	(0101)	
4	(211)	(0110)	
5	(220)	(0200)	
6		(1001)	
7		(1010)	
8		(1100)	
9		(2000)	

$$\begin{aligned}&= \sum_{j=0}^{5} (-1)^j \binom{5}{j}\binom{5-j3-1}{4} \\ &= \binom{5}{0}\binom{4}{4} \\ &= 1.\end{aligned}$$

All $(1, 3)$-constrained words of length $m = 10$ and weight $w \in \{3, 4, 5\}$ are shown in Table 1. Therefore

$$\begin{aligned}B(20, 1, 3) &= R^2(7, 3; 3) + R(7, 3; 3)R(6, 4; 3) + R^2(6, 4; 3) \\ &\quad + R(6, 4; 3)R(5, 5; 3) + R^2(5, 5; 3) \\ &= 6^2 + 6(10) + 10^2 + 10(1) + 1^2 \\ &= 207,\end{aligned}$$

which agrees with the value in [9, Tab. 1]. □

Let $C(d, k)$ denote the capacity of (d, k)-constrained codes. Zehavi and Wolf [13] show that the probability distribution of run-length segment lengths of a maxentropic (d, k)-constrained sequence is such that a run-length segment of length i, $d + 1 \leq i \leq k + 1$, has a probability $2^{-iC(d,k)}$. They obtained this result using

$$C(d, k) = \lim_{n\to\infty} \sup_{P_n(X)} \frac{H(X_1, X_2, \ldots, X_n)}{E(X_1 + X_2 + \ldots + X_n)},$$

where X_i is a random variable denoting the length of the ith run-length segment, H is the entropy function, E is the expectation operator, and the supremum is taken over all joint probabilities $P_n(X)$ of the n random variables $X_1, X_2, \ldots, X_n$. By showing that the X_i are independent and identically distributed, applying Lagrange multipliers they prove that $P(X = i) = 2^{-iC(d,k)}$, $d + 1 \leq i \leq k + 1$.

Let $C_{\text{bal}}(d, k)$ denote the capacity of balanced (d, k)-constrained codes. Manada and Morita [8] prove that $C_{\text{bal}}(d, k) = C(d, k)$. This result can be more easily proved using the approach of Zehavi and Wolf. Note that

$$C_{\text{bal}}(d, k) = \lim_{n\to\infty} \sup_{P_n(X,Y)} \frac{H(X_1, Y_1, X_2, Y_2, \ldots, X_n, Y_n)}{E(X_1 + Y_1 + X_2 + Y_2 + \ldots + X_n + Y_n)},$$

where X_i is the length of the ith odd-indexed run-length segment and Y_i is the length of the ith even-indexed run-length segment, and the supremum is taken over all joint probabilities $P_n(X, Y)$ of the $2n$ random variables $X_1, Y_1, X_2, Y_2 \ldots, X_n, Y_n$. In essence, $X_1, X_2, \ldots, X_n$ and $Y_1, Y_2, \ldots, Y_n$ represent the run-lengths of the two interleaved RLL subwords. For the resultant word to be balanced, it is required that $E(X_1+X_2+\ldots+X_n) = E(Y_1+Y_2+\ldots+Y_n)$. Since the two RLL subwords represented by $X_1, X_2, \ldots, X_n$ and $Y_1, Y_2, \ldots, Y_n$ are independent, it follows that

$$\begin{aligned}&C_{\text{bal}}(d, k) \\ &= \lim_{n\to\infty} \sup_{P_n(X),P_n(Y)} \frac{H(X_1, X_2, \ldots, X_n) + H(Y_1, Y_2, \ldots, Y_n)}{E(X_1 + \ldots + X_n) + E(Y_1 + \ldots + Y_n)} \\ &\leq \lim_{n\to\infty} \max\left\{ \sup_{P_n(X)} \frac{H(X_1, X_2, \ldots, X_n)}{E(X_1 + X_2 + \ldots + X_n)}, \right. \\ &\quad \left. \sup_{P_n(Y)} \frac{H(Y_1, Y_2, \ldots, Y_n)}{E(Y_1 + Y_2 + \ldots + Y_n)} \right\}, \qquad (2)\end{aligned}$$

where the inequality follows since

$$\frac{A + B}{C + D} \leq \max\left\{\frac{A}{C}, \frac{B}{D}\right\}, \quad A, B, C, D > 0.$$

Therefore, to maximize $C_{\text{bal}}(d, k)$, either term in (2) needs to be maximized. But, since both terms correspond to $C(d, k)$, it follows that $C_{\text{bal}}(d, k) = C(d, k)$. It is worth noting that both X and Y should have the same distributions to ensure that $E(X) = E(Y)$. Therefore, $P(X = i) = P(Y = i) = 2^{-iC(d,k)}$. This gives the run-length segment length distribution of the interleaved subwords for maxentropic balanced RLL sequences, a result that will be leveraged in the next section.

IV. QUASI-ENUMERATIVE CODING

Enumerative coding for constrained codes has the distinctive appeal that the resultant codes are optimal in terms of code rate as all possible constrained codewords are included in the codebook. The concept of enumerative coding was originally proposed by Cover [14]. Enumerative coding requires a lexicographic ordering or ranking of codewords. For two arbitrary codewords $\boldsymbol{\rho}^{(1)} = (\rho_1^{(1)}\rho_2^{(1)}\ldots\rho_w^{(1)})$ and $\boldsymbol{\rho}^{(2)} = (\rho_1^{(2)}\rho_2^{(2)}\ldots\rho_w^{(2)})$, $\boldsymbol{\rho}^{(1)}$ lexicographically precedes $\boldsymbol{\rho}^{(2)}$ (or $\boldsymbol{\rho}^{(1)}$ has a lower rank than $\boldsymbol{\rho}^{(2)}$), if $\rho_j^{(1)} < \rho_j^{(2)}$ and $\rho_i^{(1)} = \rho_i^{(2)}$ $\forall\, i < j$. Let $\mathcal{B} \subseteq \{0, 1, \ldots, r-1\}^w$ be a codebook consisting of $|\mathcal{B}|$ codewords. Enumerative coding is a bijective mapping

$$l : \{0, 1, 2, \ldots, |\mathcal{B}| - 1\} \to \mathcal{B}$$

between the lexicographic index or rank and the corresponding codeword $\boldsymbol{\rho} = (\rho_1\rho_2\ldots\rho_w) \in \mathcal{B}$. If $N(\rho_1\rho_2\ldots\rho_i)$ represents the number of codewords in $\mathcal{B}$ whose first i elements are $(\rho_1\rho_2\ldots\rho_i)$, then the lexicographic index or rank of an arbitrary $\boldsymbol{\rho} \in \mathcal{B}$ is

$$l^{-1}(\boldsymbol{\rho}) = \sum_{i=1}^{w} \sum_{j=0}^{\rho_j - 1} N(\rho_1\rho_2\ldots\rho_{i-1}j).$$

Kurmaev [7] presents an enumerative coding technique for balanced RLL codes. It operates directly on the binary

bit-level and determines N at any index based on the running digital sum or charge of the bits up to that index and the required charge "unbalance" which needs to be compensated for by the remaining bits. Since the calculation of N needs to be done at each index and the run-length boundaries and charge "unbalance" change at each index, it is computationally intensive, especially for large codeword lengths. An advantage of Kurmaev's approach is that it caters for constraints on the leading and trailing runs of zeros, allowing a run to transverse codeword boundaries through the use of merging bits.

The general enumerative coding technique developed by Hareedy et al. [15] cannot be applied to balanced RLL codes since balanced RLL codes cannot be defined by means of a finite set of forbidden subwords, nor can it be applied to RLL words consisting of a fixed number of run-length segments.

For encoding and decoding purposes, what is paramount is that the mapping between rank and codeword be bijective, but there is no real requirement that the ranking has to follow a strict lexicographic ordering of the codewords. In fact, dispensing with strict lexicographic ordering allows us to construct more efficient enumerative coding techniques for balanced RLL codes. We will call such enumerative coding, where it need not follow strict lexicographic ordering, *quasi-enumerative coding*.

Definition 2: Quasi-enumerative coding is a form of enumerative coding where the mapping between rank and codeword is bijective, but the rank need not correspond to a lexicographic ordering of the codewords.

The utility of quasi-enumerative coding in the context of balanced RLL codes becomes immediately apparent. Since balanced RLL words are created by interleaving two RLL subwords of a particular number of run-length segments, we can group the balanced RLL codewords based on the number of run-length segments of the constituent interleaved RLL subwords. Within each group, the ordering of the RLL subwords will follow lexicographic ordering, but the resultant overall ranking will not be lexicographically ordered.

A. CONSTRUCTION 1

This first proposed construction enumerates all the possible balanced (d, k)-constrained words of length $2m$ where each run-length segment in the word satisfies the particular (d, k)-constraint. The grouping is defined by the number of runs of the possible (d, k)-constrained interleaved subwords. The first group consists of words where both subwords consist of $w_{\min}$ runs, the second group of words where one subword consists of $w_{\min}+1$ runs and the other consists of $w_{\min}$ runs, etc. If the rank l of a word is based on this group ordering, then the group (specified by $w^{(1)}$ and $w^{(2)}$) is determined by the smallest integer j, $0 \le j \le 2(w_{\max}-w_{\min})$, such that

$$\sum_{i=0}^{j} \hat{R}(i) \ge l+1,$$

where $w^{(1)} = w_{\min} + \lceil j/2 \rceil$ and $w^{(2)} = w^{(1)} - i \bmod 2$ and $\hat{R}(i) \triangleq R^{(1)}(i)R^{(2)}(i)$, where $R^{(1)}(i) \triangleq R(m - w^{(1)}d, w^{(1)}; r)$ and $R^{(2)}(i) \triangleq R(m - w^{(2)}d, w^{(2)}; r)$. Note that

$$B(2m, d, k) = \sum_{i=0}^{2(w_{\max}-w_{\min})} \hat{R}(i).$$

The group corresponding to j has $w^{(1)} = w_{\min} + \lceil j/2 \rceil$ and $w^{(2)} = w^{(1)} - j \bmod 2$. Having identified the group to which rank l belongs, we need to find the rank within the group. This is given by the *residual rank* $\hat{l}$, where

$$\hat{l} = l - \sum_{i=0}^{j-1} \hat{R}(i).$$

Since the group corresponding to j contains $\hat{R}(j) = R^{(1)}(j)R^{(2)}(j)$ words, it follows that $\hat{l} \in \{0, 1, \ldots, \hat{R}(j)-1\}$. The residual rank $\hat{l}$ is decomposed into *partial ranks* $\hat{l}^{(1)} = \lfloor \hat{l}/R^{(2)} \rfloor$ and $\hat{l}^{(2)} = \hat{l} \bmod R^{(2)}$, so that $\hat{l} = \hat{l}^{(1)}R^{(2)} + \hat{l}^{(2)}$. The partial ranks $\hat{l}^{(1)} \in \{0, 1, \ldots, R^{(1)}-1\}$ and $\hat{l}^{(2)} \in \{0, 1, \ldots, R^{(2)}-1\}$ correspond to the lexicographic order of (d, k)-constrained subwords of length m consisting of $w^{(1)}$ and $w^{(2)}$ run-length segments, respectively.

Suppose that $\mathcal{R}_{(d,k)}(m, w)$ is the set of all (d, k)-constrained words of length m consisting of w run-length segments. We wish to develop an enumerative coding technique for this set. Note that $|\mathcal{R}_{(d,k)}(m, w)| = R(m - wd, w; r = k-d+1)$. Since $R(n-i, w-1; r)$ is the number of restricted compositions of n that start with $i \in \{1, 2, \ldots, r\}$, it follows that

$$R(n, w; r) = \sum_{i=1}^{r} R(n-i, w-1; r).$$

Given an arbitrary rank (or lexicographic index) $\hat{l} \in \{0, 1, \ldots, R(m - wd, w; r) - 1\}$, the corresponding word $\boldsymbol{\rho} = (\rho_1 \rho_2 \ldots \rho_w) \in \mathcal{R}_{(d,k)}(m, w) \subset \{0, 1, \ldots, r-1\}^w$ can be determined as follows. Initialize $n = m - wd$ and for each index $i = 1, 2, \ldots, w-1$:

1) Find the smallest integer j, $1 \le j \le r$, such that
$$\sum_{i=1}^{j} R(n-i, w-1; r) \ge \hat{l}+1.$$
Then $\rho_i = j-1$.
2) Set $\hat{l} \leftarrow \hat{l} - \sum_{i=1}^{j-1} R(n-i, w-1; r)$, $n \leftarrow n-j$ and $w \leftarrow w-1$.

Finally, $\rho_w = m - wd - 1 - \sum_{i=1}^{w-1} \rho_i + 1$. Note that the rank (or lexicographic ordering) of $\boldsymbol{\rho}$ corresponds to the inverse rank (or lexicographic ordering) of the corresponding (d, k)-constrained binary word $\boldsymbol{x}$.

Then, the subwords $\boldsymbol{\rho}^{(1)}$ and $\boldsymbol{\rho}^{(2)}$ corresponding to the partial ranks $\hat{l}^{(1)}$ and $\hat{l}^{(2)}$, respectively, can be determined using this enumerative algorithm. Here, the desired balanced RLL word is $\boldsymbol{\rho} \equiv \langle \boldsymbol{\rho}^{(1)}, \boldsymbol{\rho}^{(2)} \rangle$.

Conversely, suppose we want to find the rank $\hat{l}^{-1}(\boldsymbol{\rho})$ of an arbitrary (d, k)-constrained word $\boldsymbol{\rho} \in \mathcal{R}_{(d,k)}(m, w)$. Then

$$\hat{l}^{-1}(\boldsymbol{\rho}) = \sum_{i=1}^{w-1} \sum_{j=1}^{\rho_i} R\Big(m - wd - j - \sum_{h=1}^{i-1} \rho_h + 1, w-i; r\Big).$$

TABLE 2. Grouping by weights $w^{(1)}$ and $w^{(2)}$.

Group	i	$w^{(1)}$	$w^{(2)}$
1	0	3	3
2	1	4	3
3	2	4	4
4	3	5	4
5	4	5	5

Furthermore, suppose we want to find the rank $l^{-1}(\boldsymbol{\rho})$ of an arbitrary balanced RLL word $\boldsymbol{\rho}$. First, deinterleave $\boldsymbol{\rho}$ as $\langle \boldsymbol{\rho}^{(1)}, \boldsymbol{\rho}^{(2)} \rangle \equiv \boldsymbol{\rho}$. Let $w^{(1)}$ and $w^{(2)}$ be the number of run-length segments in $\boldsymbol{\rho}^{(1)}$ and $\boldsymbol{\rho}^{(2)}$, respectively. Then

$$l^{-1}(\boldsymbol{\rho}) = \hat{l}^{-1}(\boldsymbol{\rho}^{(1)})R^{(2)} + \hat{l}^{-1}(\boldsymbol{\rho}^{(2)}) + \sum_{i=0}^{j-1} \hat{R}(i),$$

where $j = 2(w^{(1)} - w_{\min}) + w^{(2)} - w^{(1)}$.

Example 3: We continue with the parameters from Example 2, where $(1, 3)$-constrained words of length 20 ($m = 10$) were considered. Recall that $3 \leq w \leq 5$. The constrained words are grouped according to the weights $w^{(1)}$ and $w^{(2)}$ of the constituent constrained subwords. In this example, there are five possible groups, which are shown in Table 2.

Suppose we want to find the $(1, 3)$-constrained word of rank $l = 66$. Since $R^2(7, 3; 3) = 6^2 = 36 < 66+1 = 67$ and $R^2(7, 3; 3) + R(7, 3; 3)R(6, 4; 3) = 6^2 + 6(10) = 96 \geq 67$, it follows that $j = 1$ and $w^{(1)} = 3 + \lceil 1/2 \rceil = 4$ and $w^{(2)} = 4 - 1 \bmod 2 = 3$. The residual rank is $\hat{l} = l - R^2(7, 3; 3) = 66 - 36 = 30$. Then, if $R^{(1)} \triangleq R(6, 4; 3)$ and $R^{(2)} \triangleq R(7, 3; 3)$, the partial ranks are $\hat{l}^{(1)} = \lfloor \hat{l}/R^{(2)} \rfloor = \lfloor 30/6 \rfloor = 5$ and $\hat{l}^{(2)} = \hat{l} \bmod R^{(2)} = 30 \bmod 6 = 0$, so that $\hat{l} = \hat{l}^{(1)}R^{(2)} + \hat{l}^{(2)} = 5(6) + 0$.

Now, we need to find $\boldsymbol{\rho}^{(1)}$ consisting of $w^{(1)} = 4$ runs corresponding to partial rank $\hat{l}^{(1)} = 5$ and $\boldsymbol{\rho}^{(2)}$ consisting of $w^{(2)} = 3$ runs corresponding to partial rank $\hat{l}^{(2)} = 0$. First, to find $\boldsymbol{\rho}^{(1)}$, initialize $n^{(1)} = m - w^{(1)}d = 6$, and then:

- $i = 1$:
  1) $R(5, 3; 3) = 6 \geq 5 + 1 = 6$ and so $j = 1$ and $\rho_1^{(1)} = j - 1 = 0$.
  2) $\hat{l}^{(1)} \leftarrow \hat{l}^{(1)} - \sum_{i=1}^{0} R(6 - i, 3; 3) = 5$, $n^{(1)} \leftarrow 6 - 1 = 5$ and $w^{(1)} \leftarrow w^{(1)} - 1 = 3$.
- $i = 2$:
  1) $R(4, 2; 3) + R(3, 2; 3) + R(2, 2; 3) = 3 + 2 + 1 = 6 \geq 5 + 1 = 6$ and so $j = 3$ and $\rho_2^{(1)} = j - 1 = 2$.
  2) $\hat{l}^{(1)} \leftarrow \hat{l}^{(1)} - \sum_{i=1}^{2} R(5 - i, 2; 3) = 0$, $n^{(1)} \leftarrow 5 - 3 = 2$ and $w^{(1)} \leftarrow w^{(1)} - 1 = 2$.
- $i = 3$:
  1) $R(1, 1; 3) = 1 \geq 0 + 1 = 1$ and so $j = 1$ and $\rho_3^{(1)} = j - 1 = 0$.

Finally, $\rho_4^{(1)} = 6 - 1 - \sum_{i=1}^{3} \rho_i^{(1)} + 1 = 5 - (1 + 3 + 1) = 0$. Hence, $\boldsymbol{\rho}^{(1)} = (0200)$. Similarly, to find $\boldsymbol{\rho}^{(2)}$, initialize $n^{(2)} = m - w^{(2)}d = 7$, and then:

- $i = 1$:
  1) $R(6, 2; 3) = 1 \geq 0 + 1 = 1$ and so $j = 1$ and $\rho_1^{(2)} = j - 1 = 0$.
  2) $\hat{l}^{(2)} \leftarrow \hat{l}^{(2)} - \sum_{i=1}^{0} R(7 - i, 2; 3) = 0$, $n^{(2)} \leftarrow 7 - 1 = 6$ and $w^{(2)} \leftarrow w^{(2)} - 1 = 2$.
- $i = 2$:
  1) $R(5, 1; 3) + R(4, 1; 3) + R(3, 1; 3) = 0 + 0 + 1 = 1 \geq 0 + 1 = 1$ and so $j = 3$ and $\rho_2^{(2)} = j - 1 = 2$.

Finally, $\rho_3^{(2)} = 7 - 1 - \sum_{i=1}^{2} \rho_i^{(2)} + 1 = 6 - (1 + 3) = 2$. Hence, $\boldsymbol{\rho}^{(2)} = (022)$. Therefore, the balanced $(1, 3)$-constrained word of rank $l = 66$ is $\boldsymbol{\rho} = (0022020) \equiv \langle \boldsymbol{\rho}^{(1)}, \boldsymbol{\rho}^{(2)} \rangle = \langle (0200), (022) \rangle$.

Conversely, if we are given the balanced $(1, 3)$-constrained word $\boldsymbol{\rho} = (0022020)$, the rank $l^{-1}(\boldsymbol{\rho})$ can be found as follows. First, deinterleave $\boldsymbol{\rho}$ as $\langle \boldsymbol{\rho}^{(1)}, \boldsymbol{\rho}^{(2)} \rangle = \langle (0200), (022) \rangle \equiv \boldsymbol{\rho} = (0022020)$. Hence, $w^{(1)} = 4$ and $w^{(2)} = 3$. Then

$$\begin{aligned}
\hat{l}^{-1}(\boldsymbol{\rho}^{(1)}) &= \sum_{i=1}^{3} \sum_{j=1}^{\rho_i^{(1)}} R\Big(6 - j - \sum_{h=1}^{i-1} \rho_h^{(1)} + 1, 4 - i; r\Big) \\
&= \sum_{j=1}^{0} R\Big(6 - j - \sum_{h=1}^{0} \rho_h^{(1)} + 1, 3; r\Big) \\
&\quad + \sum_{j=1}^{2} R\Big(6 - j - \sum_{h=1}^{1} \rho_h^{(1)} + 1, 2; r\Big) \\
&\quad + \sum_{j=1}^{0} R\Big(6 - j - \sum_{h=1}^{2} \rho_h^{(1)} + 1, 1; r\Big) \\
&= R(4, 2; 3) + R(3, 2; 3) \\
&= 3 + 2 \\
&= 5
\end{aligned}$$

and

$$\begin{aligned}
\hat{l}^{-1}(\boldsymbol{\rho}^{(2)}) &= \sum_{i=1}^{2} \sum_{j=1}^{\rho_i^{(2)}} R\Big(7 - j - \sum_{h=1}^{i-1} \rho_h^{(2)} + 1, 3 - i; r\Big) \\
&= \sum_{j=1}^{0} R\Big(7 - j - \sum_{h=1}^{0} \rho_h^{(2)} + 1, 2; r\Big) \\
&\quad + \sum_{j=1}^{2} R\Big(7 - j - \sum_{h=1}^{1} \rho_h^{(2)} + 1, 1; r\Big) \\
&= R(5, 1; 3) + R(4, 1; 3) \\
&= 0.
\end{aligned}$$

Since $R^{(1)} = R(m - w^{(1)}d, w^{(1)}; r) = R(6, 4; 3) = 10$, $R^{(2)} = R(m - w^{(2)}d, w^{(2)}; r) = R(7, 3; 3) = 6$ and $j = 2(4 - 3) + 3 - 4 = 1$, it follows that

$$\begin{aligned}
l^{-1}(\boldsymbol{\rho}) &= \hat{l}^{-1}(\boldsymbol{\rho}^{(1)})R^{(2)} + \hat{l}^{-1}(\boldsymbol{\rho}^{(2)}) + \sum_{i=0}^{0} \hat{R}(i) \\
&= 5(6) + 0 + \hat{R}(0) \\
&= 30 + 36 \\
&= 66.
\end{aligned}$$

□

Calculating each of the terms $R(n, w; r)$ separately is inefficient. Fortunately, the parameters update in a predictable manner, which can be utilized to lower the computational complexity. It is easy to see that

$$\binom{w+1}{j} = \frac{w+1}{w+1-j}\binom{w}{j}$$

and

$$\begin{aligned}&\binom{n-d-jr-1}{w}\\ &= \frac{(n-jr-w)(n-1-jr-w)\dots(n-d-jr-w)}{w(n-jr-1)(n-jr-2)\dots(n-jr-d)}\\ &\quad\times\binom{n-jr-1}{w-1},\end{aligned}$$

and hence

$$\begin{aligned}&\binom{w+1}{j}\binom{n-d-jr-1}{w} - \binom{w}{j}\binom{n-jr-1}{w-1}\\ &= \left(\frac{(n-jr-w)(n-1-jr-w)\dots(n-d-jr-w)}{w(n-jr-1)(n-jr-2)\dots(n-jr-d)}\right.\\ &\quad\left.\times\frac{w+1}{w+1-j} - 1\right)\binom{w}{j}\binom{n-jr-1}{w-1}.\end{aligned}$$

Also

$$\binom{w-1}{j} = \frac{w-j}{w}\binom{w}{j}$$

and

$$\binom{n-1-jr-1}{w-2} = \frac{w-1}{n-jr-1}\binom{n-jr-1}{w-1}$$

and hence

$$\begin{aligned}&\binom{w-1}{j}\binom{n-1-jr-1}{w-2} - \binom{w}{j}\binom{n-jr-1}{w-1}\\ &= \left(\frac{w-j}{w}\frac{w-1}{n-jr-1} - 1\right)\binom{w}{j}\binom{n-jr-1}{w-1}.\end{aligned}$$

Finally

$$\binom{n-1-jr-1}{w-1} = \frac{n-jr-w}{n-jr-1}\binom{n-jr-1}{w-1}$$

and hence

$$\begin{aligned}&\binom{n-1-jr-1}{w} - \binom{n-jr-1}{w}\\ &= \left(\frac{n-jr-1-w}{n-jr-1} - 1\right)\binom{n-jr-1}{w}.\end{aligned}$$

The quasi-enumerative encoding of Construction 1 is described by Algorithms 1, 2 and 3, while the corresponding quasi-enumerative decoding is described by Algorithms 4, 5 and 6.

Algorithm 1 Encoding of Construction 1

Input: Rank l, (d, k)-constraint, word length $2m$
Output: Balanced (d, k)-constrained word $\boldsymbol{\rho}$

EncodeCon1$(l, d, k, 2m)$
  $r \leftarrow k - d + 1$
  $w \leftarrow \lceil \frac{m}{k+1} \rceil$
  $n \leftarrow m - wd$
  $\hat{l}^{(1)}, \hat{l}^{(2)}, w^{(1)}, w^{(2)}, R^{(1)}, R^{(2)}, \boldsymbol{\alpha}^{(1)}, \boldsymbol{\alpha}^{(2)} \leftarrow$
    GroupEncodeCon1(l, n, w, r, d)
  $\boldsymbol{\rho}^{(1)} \leftarrow$ RLLEncode$(\hat{l}^{(1)}, m, w^{(1)}, d, r, R^{(1)}, \boldsymbol{\alpha}^{(1)})$
  $\boldsymbol{\rho}^{(2)} \leftarrow$ RLLEncode$(\hat{l}^{(2)}, m, w^{(2)}, d, r, R^{(2)}, \boldsymbol{\alpha}^{(2)})$
  return $\langle \boldsymbol{\rho}^{(1)}, \boldsymbol{\rho}^{(2)} \rangle$

Algorithm 2 Finding the Group ($w^{(1)}$ and $w^{(2)}$)

GroupEncodeCon1(l, n, w, r, d)
  $i \leftarrow -1, \lambda \leftarrow 0, j_{\max} \leftarrow \lfloor \frac{n-w}{r} \rfloor$
  $\boldsymbol{\alpha}^{(1)} \leftarrow (\alpha_0^{(1)}\alpha_1^{(1)}\dots\alpha_{j_{\max}}^{(1)})$
  $\boldsymbol{\alpha}^{(2)} \leftarrow (\alpha_0^{(2)}\alpha_1^{(2)}\dots\alpha_{j_{\max}}^{(2)})$
  $\alpha_j^{(1)} \leftarrow (-1)^j\binom{w}{j}\binom{n-jr-1}{w-1}$ **for** $j = 0, 1, \dots, j_{\max}$
  $R^{(1)} \leftarrow \sum_{j=0}^{j_{\max}} \alpha_j^{(1)}$
  while $\lambda < l + 1$
    $i \leftarrow i + 1$
    if $i \bmod 2 = 0$
      $R^{(2)} \leftarrow R^{(1)}, \boldsymbol{\alpha}^{(2)} \leftarrow \boldsymbol{\alpha}^{(1)}$
      $\lambda_{\text{prev}} \leftarrow \lambda, \lambda \leftarrow \lambda + (R^{(1)})^2$
    else
      $j'_{\max} \leftarrow j_{\max}, j_{\max} \leftarrow \lfloor \frac{n-d-(w+1)}{r} \rfloor$
      $\delta_j \leftarrow \left(\frac{w+1}{w+1-j}\frac{\sum_{i=0}^{d} n-i-jr-w}{w\sum_{i=1}^{d} n-i-jr} - 1\right)\alpha_j^{(1)}$
        for $j = 0, 1, \dots, j_{\max}$
      $\alpha_j^{(1)} \leftarrow \alpha_j^{(1)} + \delta_j$ **for** $j = 0, 1, \dots, j_{\max}$
      $R^{(1)} \leftarrow R^{(1)} + \sum_{j=0}^{j_{\max}} \delta_j - \sum_{j=j_{\max}+1}^{j'_{\max}} \alpha_j^{(1)}$
      $\lambda_{\text{prev}} \leftarrow \lambda, \lambda \leftarrow \lambda + R^{(1)}R^{(2)}$
      $w \leftarrow w + 1, n \leftarrow n - d$
  $\hat{l} \leftarrow l - \lambda_{\text{prev}}$
  if $i \bmod 2 = 0$
    return $\lfloor \frac{\hat{l}}{R^{(2)}} \rfloor, \hat{l} \bmod R^{(2)}, w, w, R^{(1)}, R^{(2)},$
      $\boldsymbol{\alpha}^{(1)}, \boldsymbol{\alpha}^{(2)}$
  else
    return $\lfloor \frac{\hat{l}}{R^{(2)}} \rfloor, \hat{l} \bmod R^{(2)}, w, w - 1, R^{(1)}, R^{(2)},$
      $\boldsymbol{\alpha}^{(1)}, \boldsymbol{\alpha}^{(2)}$

B. CONSTRUCTION 2

This and the following quasi-enumerative constructions are based on permutation RLL codes introduced by Datta and McLaughlin [16]. They also present enumerative encoding and decoding algorithms for permutation RLL codes. Permutation RLL codes consist of codewords $\boldsymbol{\rho} = (\rho_1\rho_2\dots\rho_w)$ consisting of w run-length segments where $\rho_i \in \{0, 1, \dots, r-1\}$, $1 \le i \le w$, and $r = k - d + 1$, i.e., the ith run-length segment of length $\rho_i'', d + 1 \le \rho_i'' \le k + 1$,

Algorithm 3 Encoding of $\mathcal{R}_{(d,k)}(m, w)$

RLLEncode($\hat{l}, m, w, d, r, R, \boldsymbol{\alpha}$)
  $\boldsymbol{\rho} \leftarrow (\rho_1 \rho_2 \dots \rho_w)$
  $n \leftarrow m - wd, j_{\max} \leftarrow \lfloor \frac{n-w}{r} \rfloor$
  $w' \leftarrow w, t \leftarrow 0$
  for $i = 1, 2, \dots, w-1$
    $s \leftarrow -1, \lambda \leftarrow 0$
    while $\lambda < \hat{l} + 1$
      $s \leftarrow s + 1$
      if $s = 0$
        $j'_{\max} \leftarrow j_{\max}, j_{\max} \leftarrow \lfloor \frac{n-w'}{r} \rfloor$
        $\delta_j \leftarrow (\frac{w'-j}{w'} \frac{w'-1}{n-jr-1} - 1)\alpha_j$
          for $j = 0, 1, \dots, j_{\max}$
        $w' \leftarrow w' - 1$
      else
        $j'_{\max} \leftarrow j_{\max}, j_{\max} \leftarrow \lfloor \frac{n-1-w'}{r} \rfloor$
        $\delta_j \leftarrow (\frac{n-jr-w'}{n-jr-1} - 1)\alpha_j$ **for** $j = 0, 1, \dots, j_{\max}$
      $\alpha_j \leftarrow \alpha_j + \delta_j$ **for** $j = 0, 1, \dots, j_{\max}$
      $R \leftarrow R + \sum_{j=0}^{j_{\max}} \delta_j - \sum_{j=j_{\max}+1}^{j'_{\max}} \alpha_j$
      $\lambda_{\text{prev}} \leftarrow \lambda, \lambda \leftarrow \lambda + R$
      $n \leftarrow n - 1$
    $\hat{l} \leftarrow \hat{l} - \lambda_{\text{prev}}$
    $\rho_i \leftarrow s, t \leftarrow t + s + 1$
  $\rho_w \leftarrow m - wd - 1 - t$
  return $\boldsymbol{\rho}$

Algorithm 4 Decoding of Construction 1

Input: Balanced (d, k)-constrained word $\boldsymbol{\rho} \equiv \langle \boldsymbol{\rho}^{(1)}, \boldsymbol{\rho}^{(2)} \rangle$, (d, k)-constraint, word length $2m$
Output: Rank l

DecodeCon1($\boldsymbol{\rho}^{(1)}, \boldsymbol{\rho}^{(2)}, w^{(1)}, w^{(2)}, d, k, 2m$)
  $r \leftarrow k - d + 1$
  $w \leftarrow \lceil \frac{m}{k+1} \rceil$
  $n \leftarrow m - wd$
  $l', R^{(1)}, R^{(2)}, \boldsymbol{\alpha}^{(1)}, \boldsymbol{\alpha}^{(2)} \leftarrow$
    GroupDecodeCon1($n, w^{(1)}, w^{(2)}, w, r, d$)
  $\hat{l}^{(1)} \leftarrow$ RLLDecode($m, \boldsymbol{\rho}^{(1)}, d, r, R^{(1)}, \boldsymbol{\alpha}^{(1)}$)
  $\hat{l}^{(2)} \leftarrow$ RLLDecode($m, \boldsymbol{\rho}^{(2)}, d, r, R^{(2)}, \boldsymbol{\alpha}^{(2)}$)
  return $l' + \hat{l}^{(1)} R^{(2)} + \hat{l}^{(2)}$

is represented by $\rho_i = \rho''_i - (d+1)$. Let ν_j, $0 \le j \le r-1$, be the number of symbols j in $\boldsymbol{\rho}$. Clearly $\sum_{j=0}^{r-1} \nu_j = w$. The length of the binary (d, k)-constrained word $\boldsymbol{x} = (x_1 x_2 \dots x_m)$ corresponding to $\boldsymbol{\rho}$ is

$$m = w(d+1) + \sum_{i=1}^{w} \rho_i = w(d+1) + \sum_{j=0}^{r-1} j\nu_j.$$

Due to the result by Zehavi and Wolf [13], the run-length segment probability distribution of a maxentropic RLL sequence is known to be $P(\rho_i = j) = 2^{-(j+d+1)C(d,k)}$ for $j \in \{0, 1, \dots, r-1\}$. By choosing $\nu_j / w \approx 2^{-(j+d+1)C(d,k)}$,

Algorithm 5 Finding Rank Contribution for the group ($w^{(1)}$ and $w^{(2)}$)

GroupDecodeCon1($n, w^{(1)}, w^{(2)}, w, r, d$)
  $i \leftarrow -1, \lambda \leftarrow 0, j_{\max} \leftarrow \lfloor \frac{n-w}{r} \rfloor$
  $\boldsymbol{\alpha}^{(1)} \leftarrow (\alpha_0^{(1)} \alpha_1^{(1)} \dots \alpha_{j_{\max}}^{(1)})$
  $\boldsymbol{\alpha}^{(2)} \leftarrow (\alpha_0^{(2)} \alpha_1^{(2)} \dots \alpha_{j_{\max}}^{(2)})$
  $\alpha_j^{(1)} \leftarrow (-1)^j \binom{w}{j} \binom{n-jr-1}{w-1}$ **for** $j = 0, 1, \dots, j_{\max}$
  $R^{(1)} \leftarrow \sum_{j=0}^{j_{\max}} \alpha_j^{(1)}$
  while $w \le w^{(1)}$
    $i \leftarrow i + 1$
    if $i \bmod 2 = 0$
      if $w \le w^{(1)}$ **and** $w \le w^{(2)}$
        $R^{(2)} \leftarrow R^{(1)}, \boldsymbol{\alpha}^{(2)} \leftarrow \boldsymbol{\alpha}^{(1)}$
        $\lambda_{\text{prev}} \leftarrow \lambda, \lambda \leftarrow \lambda + (R^{(1)})^2$
    else
      if $w + 1 \le w^{(1)}$ **and** $w \le w^{(2)}$
        $j'_{\max} \leftarrow j_{\max}, j_{\max} \leftarrow \lfloor \frac{n-d-(w+1)}{r} \rfloor$
        $\delta_j \leftarrow \left(\frac{w+1}{w+1-j} \frac{\sum_{i=0}^{d} n-i-jr-w}{w \sum_{i=1}^{d} n-i-jr} - 1 \right) \alpha_j^{(1)}$
          for $j = 0, 1, \dots, j_{\max}$
        $\alpha_j^{(1)} \leftarrow \alpha_j^{(1)} + \delta_j$ **for** $j = 0, 1, \dots, j_{\max}$
        $R^{(1)} \leftarrow R^{(1)} + \sum_{j=0}^{j_{\max}} \delta_j - \sum_{j=j_{\max}+1}^{j'_{\max}} \alpha_j^{(1)}$
        $\lambda_{\text{prev}} \leftarrow \lambda, \lambda \leftarrow \lambda + R^{(1)} R^{(2)}$
        $w \leftarrow w + 1, n \leftarrow n - d$
  return $\lambda_{\text{prev}}, R^{(1)}, R^{(2)}, \boldsymbol{\alpha}^{(1)}, \boldsymbol{\alpha}^{(2)}$

Algorithm 6 Decoding of $\mathcal{R}_{(d,k)}(m, w)$

RLLDecode($m, \boldsymbol{\rho}, d, r, R, \boldsymbol{\alpha}$)
  $n \leftarrow m - wd, j_{\max} \leftarrow \lfloor \frac{n-w}{r} \rfloor$
  $w' \leftarrow w, \hat{l} \leftarrow 0$
  for $i = 1, 2, \dots, w-1$
    $s \leftarrow 0, \lambda \leftarrow 0$
    while $s \le \rho_i$
      if $s = 0$
        $j'_{\max} \leftarrow j_{\max}, j_{\max} \leftarrow \lfloor \frac{n-w'}{r} \rfloor$
        $\delta_j \leftarrow (\frac{w'-j}{w'} \frac{w'-1}{n-jr-1} - 1)\alpha_j$
          for $j = 0, 1, \dots, j_{\max}$
        $w' \leftarrow w' - 1$
      else
        $j'_{\max} \leftarrow j_{\max}, j_{\max} \leftarrow \lfloor \frac{n-1-w'}{r} \rfloor$
        $\delta_j \leftarrow (\frac{n-jr-w'}{n-jr-1} - 1)\alpha_j$ **for** $j = 0, 1, \dots, j_{\max}$
      $\alpha_j \leftarrow \alpha_j + \delta_j$ **for** $j = 0, 1, \dots, j_{\max}$
      $R \leftarrow R + \sum_{j=0}^{j_{\max}} \delta_j + \sum_{j=j_{\max}+1}^{j'_{\max}} \alpha_j$
      $\lambda_{\text{prev}} \leftarrow \lambda, \lambda \leftarrow \lambda + R$
      $n \leftarrow n - 1, s \leftarrow s + 1$
    $\hat{l} \leftarrow \hat{l} + \lambda_{\text{prev}}$
  return $\hat{l}$

the rate of the permutation RLL code can be made arbitrarily close to capacity for increasing codeword length. Any permutation of $\boldsymbol{\rho}$ is a valid codeword as it also satisfies the desired run-length segment probability distribution. The

number of codewords in the permutation RLL code is

$$\frac{w!}{\nu_0!\nu_1!\dots\nu_{r-1}!}.$$

A more efficient enumerative encoding and decoding algorithm for permutation RLL codes is presented by Milenkovic and Vasić [10]. The proposed quasi-enumerative constructions will make use of this enumerative coding of permutation RLL codes.

As in Construction 1 above, a balanced RLL word is obtained by interleaving two RLL subwords consisting of a fixed number of run-length segments. The RLL subwords are from permutation RLL codes of appropriate parameters. As before, $w_{\min} = \lceil \frac{m}{k+1} \rceil$ and $w_{\max} = \lfloor \frac{m}{d+1} \rfloor$. Associate the index i, $0 \le i \le w_{\max} - w_{\min}$, to $w_{\min} + i$. Then

$$P^{(i)} \triangleq \frac{(w_{\min}+i)!}{\nu_0^{(i)}!\nu_1^{(i)}!\dots\nu_{r-1}^{(i)}!},$$

where $P \equiv P^{(0)}$. For each i, select $\nu_j^{(i)}$ so that $\nu_j^{(i)}/(w_{\min}+i)$ is as close to $2^{-(j+d+1)C(d,k)}$ as possible, i.e., maximize the number of permutation RLL words, under the constraint

$$m - (w_{\min}+i)(d+1) = \sum_{j=0}^{r-1} j\nu_j^{(i)}.$$

If

$$\nu^{(i)} \triangleq \frac{\nu_0^{(i)}!\nu_1^{(i)}!\dots\nu_{r-1}^{(i)}!}{\nu_0^{(i-1)}!\nu_1^{(i-1)}!\dots\nu_{r-1}^{(i-1)}!}$$

for $1 \le i \le w_{\max} - w_{\min}$, then it follows that

$$\begin{aligned}
P^{(1)} &= \frac{w_{\min}+1}{\nu^{(1)}}P \\
P^{(2)} &= \frac{w_{\min}+2}{\nu^{(2)}}P^{(1)} \\
&= \frac{(w_{\min}+1)(w_{\min}+2)}{\nu^{(1)}\nu^{(2)}}P \\
&\vdots \\
P^{(i)} &= \frac{w_{\min}+i}{\nu^{(i)}}P^{(i-1)} \\
&= \frac{\prod_{j=1}^{i} w_{\min}+j}{\prod_{j=1}^{i}\nu^{(j)}}P.
\end{aligned}$$

Let $B_{\text{perm}}(2m,d,k)$ represent the number of balanced (d,k)-constrained words where the subwords are permutation RLL words. Then

$$\begin{aligned}
&B_{\text{perm}}(2m,d,k) \\
&= P^2 + \sum_{i=1}^{w_{\max}-w_{\min}} P^{(i-1)}P^{(i)} + (P^{(i)})^2 \\
&= P^2 + \sum_{i=1}^{w_{\max}-w_{\min}} \left[\frac{\prod_{j=1}^{i-1} w_{\min}+j}{\prod_{j=1}^{i-1}\nu^{(j)}}\right]P\left[\frac{\prod_{j=1}^{i} w_{\min}+j}{\prod_{j=1}^{i}\nu^{(j)}}\right]P \\
&\quad + \left[\frac{\prod_{j=1}^{i} w_{\min}+j}{\prod_{j=1}^{i}\nu^{(j)}}\right]^2 P^2 \\
&= P^2\left[1 + \sum_{i=1}^{w_{\max}-w_{\min}} \left[\frac{\prod_{j=1}^{i} w_{\min}+j}{\prod_{j=1}^{i}\nu^{(j)}}\right]^2 \left(\frac{\nu^{(i)}}{w_{\min}+i}+1\right)\right].
\end{aligned}$$

Suppose we want to find a balanced RLL word of rank l. To find $w^{(1)}$ and $w^{(2)}$ of the two subwords, find the smallest j, $0 \le j \le w_{\max} - w_{\min}$, such that

$$1 + \sum_{i=1}^{j}\left[\frac{\prod_{a=1}^{i} w_{\min}+a}{\prod_{a=1}^{i}\nu^{(a)}}\right]^2\left(\frac{\nu^{(i)}}{w_{\min}+i}+1\right) \ge \frac{l+1}{P^2}.$$

Then $w^{(1)} = w_{\min} + j$. If

$$\begin{aligned}
&1 + \sum_{i=1}^{j}\left[\frac{\prod_{a=1}^{i} w_{\min}+a}{\prod_{a=1}^{i}\nu^{(a)}}\right]^2\left(\frac{\nu^{(i)}}{w_{\min}+i}+1\right) \\
&\ge \frac{l+1}{P^2} + \left[\frac{\prod_{a=1}^{j} w_{\min}+a}{\prod_{a=1}^{j}\nu^{(a)}}\right]^2
\end{aligned}$$

then $w^{(2)} = w_{\min} + j - 1$, otherwise $w^{(2)} = w_{\min} + j$. If $j = 0$, then the residual rank is $\hat{l} = l$ and the partial ranks are $\hat{l}^{(1)} = \lfloor \hat{l}/P^{(0)} \rfloor$ and $\hat{l}^{(2)} = \hat{l} \bmod P^{(0)}$, so that $\hat{l} = \hat{l}^{(1)}P^{(0)} + \hat{l}^{(2)}$. If $j \ne 0$, if $w^{(1)} = w^{(2)}$ then

$$\begin{aligned}
\hat{l} = l - P^2\Bigg[&1 - \left[\frac{\prod_{a=1}^{j} w_{\min}+a}{\prod_{a=1}^{j}\nu^{(a)}}\right]^2 \\
&+ \sum_{i=1}^{j}\left[\frac{\prod_{a=1}^{i} w_{\min}+a}{\prod_{a=1}^{i}\nu^{(a)}}\right]^2\left(\frac{\nu^{(i)}}{w_{\min}+i}+1\right)\Bigg]
\end{aligned}$$

and the partial ranks are $\hat{l}^{(1)} = \lfloor \hat{l}/P^{(j)} \rfloor$ and $\hat{l}^{(2)} = \hat{l} \bmod P^{(j)}$, so that $\hat{l} = \hat{l}^{(1)}P^{(j)} + \hat{l}^{(2)}$. Else if $w^{(1)} = w^{(2)} + 1$, then

$$\hat{l} = l - P^2\left[1 + \sum_{i=1}^{j-1}\left[\frac{\prod_{a=1}^{i} w_{\min}+a}{\prod_{a=1}^{i}\nu^{(a)}}\right]^2\left(\frac{\nu^{(i)}}{w_{\min}+i}+1\right)\right]$$

and the partial ranks are $\hat{l}^{(1)} = \lfloor \hat{l}/P^{(j-1)} \rfloor$ and $\hat{l}^{(2)} = \hat{l} \bmod P^{(j-1)}$, so that $\hat{l} = \hat{l}^{(1)}P^{(j-1)} + \hat{l}^{(2)}$.

Example 4: Consider the same parameters as from Example 3. We want to find the balanced $(1,3)$-constrained word of rank $l = 55$. Recall that $3 \le w \le 5$. Then

- $w = 3$ $(i = 0)$: $m - w(d+1) = 10 - 3(2) = 4$ and $\nu_0^{(0)} = 0$, $\nu_1^{(0)} = 2$ and $\nu_2^{(0)} = 1$. Note that $\sum_{j=0}^{2} j\nu_j^{(0)} = 4$. Then
$$P \equiv P^{(0)} = \frac{3!}{0!2!1!} = 3,$$
and the three words in lexicographical order are (112), (121) and (211).
- $w = 4$ $(i = 1)$: $m - w(d+1) = 10 - 4(2) = 2$ and $\nu_0^{(1)} = 2$, $\nu_1^{(1)} = 2$ and $\nu_2^{(1)} = 0$. Note that $\sum_{j=0}^{2} j\nu_j^{(1)} = 2$. Then
$$P^{(1)} = \frac{4!}{2!2!0!} = 6,$$
and the six words in lexicographical order are (0011), (0101), (0110), (1001), (1010) and (1100). Also
$$\nu^{(1)} = \frac{2!2!0!}{0!2!1!} = 2.$$

- $w = 5$ ($i = 2$): $m - w(d+1) = 10 - 5(2) = 0$ and $\nu_0^{(2)} = 5$, $\nu_1^{(2)} = 0$ and $\nu_2^{(2)} = 0$. Note that $\sum_{j=0}^{2} j\,\nu_j^{(2)} = 0$. Then

$$P^{(2)} = \frac{5!}{5!0!0!} = 1,$$

and the single word is (00000). Also

$$\nu^{(2)} = \frac{5!0!0!}{2!2!0!} = 30.$$

Therefore

$$\begin{aligned} &B_{\text{perm}}(20,1,3) \\ &= P^2\left[1 + \sum_{i=1}^{2}\left[\frac{\prod_{j=1}^{i} w_{\min}+j}{\prod_{j=1}^{i}\nu^{(j)}}\right]^2\left(\frac{\nu^{(i)}}{w_{\min}+i}+1\right)\right] \\ &= 9\left[1 + \left(\frac{4}{2}\right)^2\left(\frac{2}{4}+1\right) + \left(\frac{4(5)}{2(30)}\right)^2\left(\frac{30}{5}+1\right)\right] \\ &= 70 \\ &= 3^2 + 3(6) + 6^2 + 6(1) + 1^2. \end{aligned}$$

Now we can proceed to find the balanced $(1,3)$-constrained word of rank $l = 55$. For $j = 0$, $1 < (55+1)/P^2 = 56/9$. For $j = 1$

$$\begin{aligned} &1 + \sum_{i=1}^{1}\left[\frac{\prod_{j=1}^{i} w_{\min}+j}{\prod_{j=1}^{i}\nu^{(j)}}\right]^2\left(\frac{\nu^{(i)}}{w_{\min}+i}+1\right) \\ &= 1 + \left(\frac{4}{2}\right)^2\left(\frac{2}{4}+1\right) \\ &= 1 + 6 = 7 \geq 56/9. \end{aligned}$$

Hence $w^{(1)} = w_{\min} + j = 3 + 1 = 4$. Since $7 < 56/9 + 2^2$, $w^{(2)} = w^{(1)} = 4$. The residual rank is $\hat{l} = 55 - 9(1 + 6 - 4) = 28$, and so the partial ranks are $\hat{l}^{(1)} = \lfloor \hat{l}/P^{(1)} \rfloor = \lfloor 28/6 \rfloor = 4$ and $\hat{l}^{(2)} = \hat{l} \bmod P^{(1)} = 28 \bmod 6 = 4$. Then $\boldsymbol{\rho}^{(1)} = \boldsymbol{\rho}^{(2)} = (1010)$, and so

$$\begin{aligned} \boldsymbol{\rho} &\equiv \langle \boldsymbol{\rho}^{(1)}, \boldsymbol{\rho}^{(2)} \rangle \\ &= \langle (1010), (1010) \rangle \\ &= (11001100). \end{aligned}$$

□

The quasi-enumerative encoding for Construction 2 is described by Algorithms 9 and 10, while the corresponding quasi-enumerative decoding for Construction 2 is described by Algorithms 11 and 12. PermRLLEncode and PermRLLDecode represent the enumerative encoding and decoding algorithms of permutation RLL codes developed by Milenkovic and Vasić [10]. The parameters $\boldsymbol{\nu}^{(i)} = (\nu_0^{(i)}\nu_1^{(i)}\ldots\nu_{r-1}^{(i)})$ are determined algorithmically as per Algorithm 8. Since $w_{\min} + i = \sum_{j=0}^{r-1}\nu_j^{(i)}$, as i increments by one, we have

$$\sum_{j=0}^{r-1}\nu_j^{(i+1)} = 1 + \sum_{j=0}^{r-1}\nu_j^{(i)}.$$

On the other hand, $n^{(i+1)} = n^{(i)} - (d+1)$ where

$$n^{(i)} = m - (w_{\min} + i)(d+1) = \sum_{j=0}^{r-1} j\nu_j^{(i)}.$$

Since the probability of a run-length segment of length l in a maxentropic RLL sequence is $2^{-lC(d,k)}$, it follows that run-length segments of longer length are less probable. For a particular w, a heuristic algorithm to determine $\boldsymbol{\nu}^{(i)} = (\nu_0^{(i)}\nu_1^{(i)}\ldots\nu_{r-1}^{(i)})$ is as follows:

1) Set all $\nu_j^{(i)} = 0$, $0 \leq j \leq r-1$.
2) Starting at $j = 0$, for each j, find the smallest $\nu_j^{(i)}$ such that $\nu_j^{(i)}/w \geq 2^{-(d+1+j)C(d,k)}$. If $(\nu_j^{(i)} - 1)/w$ is closer to $2^{-(d+1+j)C(d,k)}$, decrement $\nu_j^{(i)}$ by one. Since $w = \sum_{j=0}^{r-1}\nu_j^{(i)}$, if $\sum_{t=j}^{r-1}\nu_t^{(i)} > w$, adjust $\nu_j^{(i)}$ accordingly.
3) If $m - (w_{\min} + i)(d+1) \neq \sum_{j=0}^{r-1} j\nu_j^{(i)}$, starting at $j = r-1$, increment ν_j by one and decrement ν_{j-1} by one to increase the length by one and vice versa to decrement the length by one. This process is repeated with decrementing j until the desired length is attained.

During the initialization stage, this algorithm is used to determine $\boldsymbol{\nu}^{(i)}$ for all $0 \leq i \leq w_{\max} - w_{\min}$. Then, all $\nu^{(i)}$, $1 \leq i \leq w_{\max} - w_{\min}$, can be calculated. This initialization step is performed only once for a particular m and (d,k) constraints and the resultant $\nu^{(i)}$ can be used for all word encoding and decoding.

Algorithm 7 Initialization of Construction 2

Input: (d,k)-constraint, word length $2m$
Output: $\boldsymbol{\nu} = (\nu^{(1)}\nu^{(2)}\ldots\nu^{(w_{\max}-w_{\min})})$

Init$(d,k,2m)$
  $r \leftarrow k - d + 1$
  $w_{\min} \leftarrow \lceil \frac{m}{k+1} \rceil$, $w_{\max} \leftarrow \lfloor \frac{m}{d+1} \rfloor$
  $\boldsymbol{\nu}^{(i)} \leftarrow (\nu_0^{(i)}\nu_1^{(i)}\ldots\nu_{r-1}^{(i)})$ **for** $i = 0, 1, \ldots, w_{\max} - w_{\min}$
  $\boldsymbol{\nu}^{(0)} \leftarrow$ FindNu$(m, w_{\min}, r, d)$
  for $i = 1, 2, \ldots, w_{\max} - w_{\min}$
    $\boldsymbol{\nu}^{(i)} \leftarrow$ FindNu$(m, w_{\min} + i, r, d)$
    $\boldsymbol{\nu}^{(i)} \leftarrow \frac{\nu_0^{(i)}!\nu_1^{(i)}!\ldots\nu_{r-1}^{(i)}!}{\nu_0^{(i-1)}!\nu_1^{(i-1)}!\ldots\nu_{r-1}^{(i-1)}!}$
  return $\boldsymbol{\nu}$

C. CONSTRUCTION 3

Construction 3 is a special case of Construction 2. Instead of considering the different possible number of run-length segments of the interleaved subwords as in Construction 2, Construction 3 only uses a single value

$$w \approx \frac{w_{\max} + w_{\min}}{2}$$

for the two interleaved subwords. This w is selected as half-way between $w_{\max}$ and $w_{\min}$ because this selection maximizes the codebook cardinality. For the selected w, the values $\nu_j/w \approx 2^{-(j+d+1)C(d,k)}$ under the constraint

Algorithm 8 Find $\boldsymbol{\nu} = (\nu_0\nu_1\ldots\nu_{r-1})$ for Particular Length m and Number of Run-Length Segments w

FindNu(m, w, r, d)
  $\nu_j \leftarrow 0$ **for** $j = 0, 1, \ldots, r-1$
  $w' \leftarrow w, m' \leftarrow m - w(d+1)$
  for $j = 0, 1, \ldots, r-2$
    $p_j \leftarrow 2^{-(d+1+j)C(d,k)}$
    while $\nu_j/w < p_j$
      $\nu_j \leftarrow \nu_j + 1$
    if $\nu_j/w - p_j > p_j - (\nu_j - 1)/w$
      $\nu_j \leftarrow \nu_j - 1$
    if $w' < \nu_j$
      $\nu_j \leftarrow w'$
    $w' \leftarrow w' - \nu_j, m' \leftarrow m' - j\nu_j$
  $\nu_{r-1} \leftarrow w', m' \leftarrow m' - (r-1)\nu_{r-1}$
  if $m' < 0$
    $j \leftarrow r-1$
    while $m' < 0$
      if $\nu_j \geq 1$
        $\nu_j \leftarrow \nu_j - 1, \nu_{j-1} \leftarrow \nu_{j-1} + 1$
        $m' \leftarrow m' + 1$
      if $j = 1$
        $j \leftarrow r-1$
      else
        $j \leftarrow j-1$
  if $m' > 0$
    $j \leftarrow r-1$
    while $m' > 0$
      if $\nu_{j-1} \geq 1$
        $\nu_j \leftarrow \nu_j + 1, \nu_{j-1} \leftarrow \nu_{j-1} - 1$
        $m' \leftarrow m' - 1$
      if $j = 1$
        $j \leftarrow r-1$
      else
        $j \leftarrow j-1$
  return $\boldsymbol{\nu}$

Algorithm 9 Encoding of Construction 2

Input: Rank l, (d, k)-constraint, word length $2m$, $\boldsymbol{\nu} = (\nu^{(1)}\nu^{(2)}\ldots\nu^{(w_{\max}-w_{\min})})$
Output: Balanced (d, k)-constrained word $\boldsymbol{\rho}$

EncodeCon2($l, d, k, 2m, \boldsymbol{\nu}$)
  $r \leftarrow k - d + 1$
  $w \leftarrow \lceil \frac{m}{k+1} \rceil$
  $\hat{l}^{(1)}, \hat{l}^{(2)}, w^{(1)}, w^{(2)} \leftarrow$
    GroupEncodeCon2($l, m, w, r, d, \boldsymbol{\nu}$)
  $\boldsymbol{\nu}^{(1)} \leftarrow$ FindNu($m, w^{(1)}, r, d$)
  $\boldsymbol{\nu}^{(2)} \leftarrow$ FindNu($m, w^{(2)}, r, d$)
  $\boldsymbol{\rho}^{(1)} \leftarrow$ PermRLLEncode($\hat{l}^{(1)}, m, w^{(1)}, d, r, \boldsymbol{\nu}^{(1)}$)
  $\boldsymbol{\rho}^{(2)} \leftarrow$ PermRLLEncode($\hat{l}^{(2)}, m, w^{(2)}, d, r, \boldsymbol{\nu}^{(2)}$)
  return $\langle \boldsymbol{\rho}^{(1)}, \boldsymbol{\rho}^{(2)} \rangle$

Algorithm 10 Finding the Group ($w^{(1)}$ and $w^{(2)}$)

GroupEncodeCon2($l, m, w, r, d, \boldsymbol{\nu}$)
  $i \leftarrow 0, \lambda \leftarrow 0, \pi \leftarrow 1$
  $\boldsymbol{\nu}^{(0)} \leftarrow$ FindNu(m, w, r, d)
  $P \leftarrow \frac{w!}{\nu_0^{(0)}!\nu_1^{(0)}!\ldots\nu_{r-1}^{(0)}!}$
  if $1 \geq \frac{l+1}{P^2}$
    return $\lfloor \frac{l}{P} \rfloor, l \bmod P, w, w$
  else
    $\lambda \leftarrow \lambda + 1$
  while $\lambda < \frac{l+1}{P^2}$
    $i \leftarrow i + 1$
    $\beta \leftarrow \frac{w+i}{\nu^{(i)}}$
    $\pi \leftarrow \pi\beta$
    $\lambda \leftarrow \lambda + \pi^2(\frac{1}{\beta} + 1)$
  if $\lambda \geq \frac{l+1}{P} + \pi^2$
    $\hat{l} \leftarrow l - P^2\lambda, P^{(i-1)} \leftarrow P\frac{\pi}{\beta}$
    return $\lfloor \frac{\hat{l}}{P^{(i-1)}} \rfloor, \hat{l} \bmod P^{(i-1)}, w+i, w+i-1$
  else
    $\hat{l} \leftarrow l - P^2\lambda + P^2\pi^2, P^{(i)} \leftarrow P\pi$
    return $\lfloor \frac{\hat{l}}{P^{(i)}} \rfloor, \hat{l} \bmod P^{(i)}, w+i, w+i$

Algorithm 11 Decoding of Construction 2

Input: Balanced (d, k)-constrained word $\boldsymbol{\rho} \equiv \langle \boldsymbol{\rho}^{(1)}, \boldsymbol{\rho}^{(2)} \rangle$, (d, k)-constraint, word length $2m$, $\boldsymbol{\nu} = (\nu^{(1)}\nu^{(2)}\ldots\nu^{(w_{\max}-w_{\min})})$
Output: Rank l

DecodeCon2($\boldsymbol{\rho}^{(1)}, \boldsymbol{\rho}^{(2)}, w^{(1)}, w^{(2)}, d, k, 2m, \boldsymbol{\nu}$)
  $r \leftarrow k - d + 1$
  $w \leftarrow \lceil \frac{m}{k+1} \rceil$
  $l', P^{(1)}, P^{(2)} \leftarrow$
    GroupDecodeCon2($m, w^{(1)}, w^{(2)}, w, r, d, \boldsymbol{\nu}$)
  $\hat{l}^{(1)} \leftarrow$ PermRLLDecode($m, \boldsymbol{\rho}^{(1)}, w^{(1)}, d, r$)
  $\hat{l}^{(2)} \leftarrow$ PermRLLDecode($m, \boldsymbol{\rho}^{(2)}, w^{(2)}, d, r$)
  return $l' + \hat{l}^{(1)}P^{(2)} + \hat{l}^{(2)}$

$w = \sum_{j=0}^{r-1} \nu_j$ are chosen in order to maximize the cardinality of the interleaved subword codebooks.

V. PERFORMANCE COMPARISON

The computational complexity and code rates of the enumerative coding by Kurmaev [7] are compared to that of Constructions 1, 2 and 3 presented above. Enumerative coding algorithms can be compared in terms of storage and/or computational requirements. The enumerative coding by Kurmaev and the quasi-enumerative constructions proposed here are by nature computationally oriented and are not well suited to the storage approach. Hence, the comparison will be in terms of computational complexity.

Algorithm 12 Finding Rank Contribution for the Group ($w^{(1)}$ and $w^{(2)}$)

GroupDecodeCon2(m, $w^{(1)}$, $w^{(2)}$, w, r, d, $\boldsymbol{\nu}$)
  $i \leftarrow 0, \lambda \leftarrow 0, \pi \leftarrow 1$
  $\boldsymbol{\nu}^{(0)} \leftarrow$ FindNu(m, w, r, d)
  $P \leftarrow \frac{w!}{\nu_0^{(0)}!\nu_1^{(0)}!\ldots\nu_{r-1}^{(0)}!}$
  if $w^{(1)} = w^{(2)} = w$
    return 0, P, P
  else
    $\lambda \leftarrow \lambda + 1, i \leftarrow i + 1$
  while $i \leq w^{(i)}$
    $\beta \leftarrow \frac{w+i}{\nu^{(i)}}$
    $\pi \leftarrow \pi\beta$
    $\lambda \leftarrow \lambda + \pi^2(\frac{1}{\beta} + 1)$
    $i \leftarrow i + 1$
  if $w^{(1)} = w^{(2)} + 1$
    return $P^2(\lambda - \pi^2(\frac{1}{\beta} + 1)), P\pi, P\frac{\pi}{\beta}$
  else
    return $P^2(\lambda - \pi^2), P\pi, P\pi$

A. COMPLEXITY

For balanced RLL codes, the crux of Kurmaev's enumerative coding is entailed in equation (16) [7, p. 4501]. Although more verbose, the similarity with (1) is obvious. This helps to facilitate the comparison. Also pertinent is equation (7) which uses (16) [7, p. 4499]. Kurmaev utilizes ($dkl'r'$) words, where first run of zeroes is at most l' and the last run of zeroes is at most r'. The comparison will be based on the order of multiplications/divisions needed since the number of multiplications/divisions exceeds the number of additions/subtractions, but it is noted that the order of additions/subtractions is the same. Rather than calculating the binomial coefficients from scratch each time, recurrence relations can be utilized as in Construction 1. Assume that the word length is $2m$ and w is the number of run-length segments. For a particular m and w, there are $4(j_{\max} + 1)$ multiplications/divisions within each summation consisting of the product of binomial coefficients, where

$$j_{\max} \approx \frac{m - w(d+1)}{r}.$$

Although this $j_{\max}$ corresponds to a charge (digital sum) $\sigma = 0$, it is noted that when the charge is non-zero, the number of summations with positive charge are approximately equal to that of the summations with the corresponding negative charge. Hence, for a particular m, the number of multiplications/divisions is approximately

$$\begin{aligned}
&(l'+1)\sum_{w=0}^{w_{\max}}\sum_{j=0}^{w} 40\frac{m - j(d+1) + r}{r} \\
&= \frac{40(l'+1)}{r}\sum_{w=0}^{w_{\max}}(w+1)m - (d+1)\frac{w(w+1)}{2} + (w+1)r \\
&= \frac{40(l'+1)}{r}\left(\left(\frac{w_{\max}(w_{\max}+1)}{2} + w_{\max} + 1\right)(m+r)\right. \\
&\quad - \frac{d+1}{2}\left(\frac{w_{\max}(w_{\max}+1)(2w_{\max}+1)}{6}\right. \\
&\quad \left.\left. + \frac{w_{\max}(w_{\max}+1)}{2}\right)\right) \\
&= \frac{40(l'+1)}{r}w_{\max}(w_{\max}+1)\left(\left(\frac{1}{2} + \frac{1}{w_{\max}}\right)(m+r)\right. \\
&\quad \left. - \frac{d+1}{2}\left(\frac{2w_{\max}+4}{6}\right)\right) \\
&\approx \frac{40(l'+1)}{r}w_{\max}(w_{\max}+1)\left(\left(\frac{1}{2} + \frac{1}{w_{\max}}\right)(m+r)\right. \\
&\quad \left. - \frac{m+2d+2}{6}\right) \\
&= O(m^3).
\end{aligned}$$

This process is repeated for each index in the word and so the total number of multiplications/divisions per word is

$$\sum_{j=1}^{2m} O(j^3) = O(m^4).$$

On the other hand, for Construction 1, the total number of multiplications/divisions to find $w^{(1)}$ and $w^{(2)}$ is at most

$$\begin{aligned}
&4\sum_{j=w_{\min}}^{w_{\max}} \frac{m - j(d+1) + r}{r} \\
&= \frac{4}{r}\left((w_{\max} - w_{\min} + 1)(m+r)\right. \\
&\quad \left. - (d+1)\left(\frac{w_{\max}(w_{\max}+1)}{2} - \frac{w_{\min}(w_{\min}-1)}{2}\right)\right) \\
&= O(m^2).
\end{aligned}$$

Similarly, to find the subwords $\boldsymbol{\rho}^{(1)}$ and $\boldsymbol{\rho}^{(2)}$, each requires at most

$$\begin{aligned}
&4r\sum_{j=0}^{w} \frac{m - j(d+1) + r}{r} \\
&= 4\left((w+1)(m+r) - (d+1)\frac{w(w+1)}{2}\right) \\
&= O(m^2)
\end{aligned}$$

multiplications/divisions. Hence, the total number of multiplications/divisions per word of Construction 1 is $O(m^2)$.

Constructions 2 and 3 are more efficient. For Construction 2, finding a particular $\boldsymbol{\nu}^{(i)}$ depends on r (which in turn depends on d and k constraints) and not on m. Therefore, the total number of multiplications/divisions to find $w^{(1)}$ and $w^{(2)}$ is at most $4O(r)(w_{\max} - w_{\min} + 1) = O(m)$. Similarly, it is easy to verify that the permutation RLL enumerative coding by Milenkovic and Vasić requires $O(m)$ multiplications/divisions. Therefore, Construction 2 is $O(m)$. Finally, it automatically follows that Construction 3 is also $O(m)$. In absolute terms, Construction 3 is the most efficient

TABLE 3. Comparison of complexities of Kurmaev's construction, Constructions 1–3 and Knuth-like constructions.

Construction	Kurmaev	1	2	3	Knuth-like
Complexity	$O(m^4)$	$O(m^2)$	$O(m)$	$O(m)$	$O(m)[O(\log m)]$

computationally of the three constructions proposed in this paper. Table 3 compares the complexities of Kurmaev's construction and Constructions 1–3. In addition, the complexity of Knuth-like balanced RLL codes (e.g. codes by Immink et al. [4] and Palunčić et al. [5]) is also included, since these codes are considered in the code rate efficiency comparison in the next subsection. These Knuth-like codes require on average $O(m)$ bit-flips during encoding and decoding. In addition, the balancing index is encoded by means of a prefix/suffix. This prefix/suffix, which also needs to satisfy certain RLL and balancing constraints, can be encoded and decoded using enumerative coding (e.g. the techniques proposed in this paper) and thus requires $O(\log m)$ multiplications/divisions.

B. CODE RATE EFFICIENCY

The code rates of the various constructions are compared for various parameters. For the enumerative code by Kurmaev, there are two options to consider. In the first option, $l' = 0$ and $r' = 0$ meaning that codeword boundaries correspond to run boundaries. Hence, the resultant code cardinality is equal to that of Construction 1. Alternatively, for non-zero l' and r', merging bits are required. It is easy to verify that merging bit constructions for RLL codewords (e.g. constructions 1 and 2 in [1, Sec. 5.4.2]) do not work for balanced RLL codewords. Merging bits

$$\underbrace{000\ldots0}_{d+1}1\underbrace{00\ldots0}_{d}$$

are applicable for balanced RLL codewords with $l' = k - d$ and $r' = k - (d + 1)$.

The comparison is in terms of efficiency

$$\eta \triangleq \frac{\bar{r}}{C(d,k)},$$

where $\bar{r}$ is the code rate. It is to be noted that efficiency is less than one for any finite codeword length since capacity by definition is when codeword length tends to infinity. Let η_K represent the efficiency of Kurmaev's code with merging bits and η_1, η_2 and η_3 represent the efficiencies of Constructions 1, 2 and 3, respectively. Then

$$\eta_K = \frac{\bar{r}_K}{C(d,k)}, \eta_1 = \frac{\bar{r}_1}{C(d,k)}, \eta_2 = \frac{\bar{r}_2}{C(d,k)}, \eta_3 = \frac{\bar{r}_3}{C(d,k)},$$

where

$$\bar{r}_K = \frac{\log_2 \hat{C}(2m-2d-2, 0, d, k, k-d, k-d-1)}{2m},$$

and

$$\bar{r}_1 = \frac{\log_2 B(2m,d,k)}{2m},$$

TABLE 4. Comparison of η_K, η_1, η_2 and η_3 for various m and (d, k) constraints.

(d,k)	$2m$	η_1	η_K	η_2	η_3
$(1,3)$	100	0.9247	0.9081	0.8737	0.8301
	200	0.9583	0.9494	0.9235	0.8912
	400	0.9770	0.9724	0.9558	0.9364
$(1,8)$	100	0.9219	0.9124	0.7783	0.7403
	200	0.9575	0.9525	0.8447	0.8136
	400	0.9770	0.9744	0.8999	0.8794
$(1,15)$	100	0.9191	0.9097	0.7646	0.7255
	200	0.9560	0.9512	0.8342	0.8082
	400	0.9762	0.9738	0.8894	0.8697
$(2,6)$	100	0.8942	0.8839	0.7572	0.7096
	200	0.9429	0.9369	0.8537	0.8254
	400	0.9692	0.9659	0.9099	0.8877
$(2,10)$	100	0.8911	0.8825	0.7110	0.6521
	200	0.9414	0.9366	0.7933	0.7669
	400	0.9685	0.9660	0.8682	0.8528
$(4,12)$	100	0.8326	0.8275	0.6168	0.5804
	200	0.9115	0.9074	0.7291	0.6979
	400	0.9529	0.9505	0.8235	0.8088

and

$$\bar{r}_2 = \frac{\log_2 B_{\text{perm}}(2m,d,k)}{2m},$$

and

$$\bar{r}_3 = \frac{\log_2 B_{\text{perm}}(2m,w,d,k)}{2m}.$$

$\hat{C}(2m,\sigma,d,k,l',r')$ is obtained with equations (7) and (16) from [7], while $B_{\text{perm}}(2m,w,d,k) = P^2$ where

$$P = \frac{w}{\nu_0!\nu_1!\ldots\nu_{r-1}!}$$

and

$$w = \left\lceil \frac{w_{\max} + w_{\min}}{2} \right\rceil.$$

Table 4 shows a comparison of η_K, η_1, η_2 and η_3. Construction 1, which is equivalent to Kurmaev's code with no merging bits and $l' = 0$ and $r' = 0$, has the best code rate. The use of merging bits with Kurmaev's code leads to a reduced code rate and efficiency. It worth noting that a better merging bits scheme could improve the code rate and efficiency. Comparatively, the reduction in code rate and efficiency of Constructions 2 and 3 is larger. This is the price of the reduced computational complexity of these two constructions.

As a final comparison, the selection of $\boldsymbol{\nu}^{(i)}$ for Constructions 2 and 3 is not limited by a particular m, but rather $\boldsymbol{\nu}^{(i)}$ is chosen algorithmically for a particular w. This will give $\boldsymbol{\nu}^{(i)}$ that is closer to the optimal run-length segment distribution. The results are shown in Table 5. A comparison of Tables 4 and 5 demonstrates that Constructions 2 and 3 can achieve similar efficiencies to that of Construction 1 at larger codeword lengths.

Thus, Constructions 1 to 3 have decreasing computational complexity and decreasing code rate efficiency. This is the fundamental trade-off between the three proposed constructions.

TABLE 5. Comparison of η_2 and η_3 for various m and (d, k) constraints where $\nu^{(i)}$ are selected for a particular w.

(d,k)	w	$2m$	η_2	η_3
$(1,3)$	50	276	0.9407	0.9232
	100	548	0.9655	0.9548
	200	1100	0.9804	0.9742
$(1,8)$	50	352	0.8924	0.8775
	100	688	0.9308	0.9252
	200	1386	0.9585	0.9555
$(1,15)$	50	352	0.8814	0.8665
	100	712	0.9223	0.9123
	200	1464	0.9509	0.9397
$(2,6)$	50	430	0.9144	0.9026
	100	864	0.9490	0.9421
	200	1738	0.9708	0.9667
$(2,10)$	50	490	0.8836	0.8733
	100	988	0.9275	0.9202
	200	1938	0.9567	0.9535
$(4,12)$	50	730	0.8818	0.8749
	100	1462	0.9286	0.9239
	150	2202	0.9475	0.9433

While Construction 1 is of maximal cardinality, Constructions 2 and 3 are not. In order to further contextualize their efficiency, it is instructive to compare them also to the efficiencies of balanced RLL codes based on Knuth-like balancing schemes. Specifically, in this context, pertinent code constructions are those of Immink et al. [4] and Palunčić et al. [5]. The codes by Palunčić et al. [5] are of variable and fixed length. The original fixed-length codes by Immink et al. [4] cannot be cascaded freely as run-length violations could occur at codeword boundaries. For fair comparison, the merging bits proposed in [5] to rectify this will be taken into account. In all these codes, the balancing procedure is applied to a RLL source word. Hence, the code rate of the overall code is dependent on the code rate of the RLL code. We will assume maximal cardinality of the RLL codes, as this is an upper bound on the desired code rates. For fixed-length codes, we compute the maximum number of RLL words of particular length. The codes by Palunčić et al. [5] require the codeword boundaries to correspond to the run-length boundaries, so the cardinalities of the fixed-length codes are computed using the generating function approach from [9]. Such a limitation does not apply for the constructions from [4]. The maximal cardinalities are between those of the above method and the cardinalities calculated using the formula in [1, Eq. (4.4)]. For the variable-length codes from [5], the cardinality is r^w, where $r = k - d + 1$ and w is the number of runs in the source RLL word (for more details, see [5, p. 7061]).

In Tables 6–8, FL1 denotes the fixed-length codes from [4], FL2 the fixed-length codes from [5], and VL1 and VL2 the variable-length codes 1 and 2 from [5], respectively. For ease of reference to Tables 5–7 in [5], the balanced RLL code length $(2m)$ is shown as the sum of the (average) source RLL word length and (average) balancing redundancy. The last column in the tables also includes η_1 for comparison purposes. For FL1, η_{Knuth} has two values separated by a dash:

TABLE 6. Comparison of η_{Knuth} for various variable- and fixed-length codes with η_2 and η_3 for $(d, k) = (1, 3)$.

$2m$	Const.	η_{Knuth}	η_2	η_3	η_1
$232\,(211+21)$	FL2	0.9024	0.9319	0.9020	0.9633
$234\,(210+24)$	FL1	$0.8904-0.8977$	0.9323	0.9038	0.9636
$300\,(279+21)$	VL1	0.8910	0.9445	0.9163	0.9706
$336\,(315+21)$	VL2	0.8982	0.9493	0.9283	0.9733
$480\,(457+23)$	FL2	0.9487	0.9616	0.9426	0.9805

TABLE 7. Comparison of η_{Knuth} for various variable- and fixed-length codes with η_2 and η_3 for $(d, k) = (1, 9)$.

$2m$	Const.	η_{Knuth}	η_2	η_3	η_1
$50\,(30+20)$	FL2	0.5674	0.6897	0.6222	0.8554
$110\,(90+20)$	VL2	0.6276	0.7780	0.7356	0.9274
$156\,(136+20)$	FL1	$0.8613-0.8679$	0.8185	0.7945	0.9466
$166\,(144+22)$	FL2	0.8577	0.8254	0.8014	0.9494
$266\,(246+20)$	VL1	0.7094	0.8680	0.8519	0.9667
$360\,(338+22)$	FL1	$0.9344-0.9372$	0.8902	0.8776	0.9745
$390\,(367+23)$	FL2	0.9368	0.8953	0.8827	0.9762
$460\,(438+22)$	VL2	0.7303	0.9058	0.8926	0.9795
$640\,(618+22)$	VL1	0.7407	0.9240	0.9136	0.9847

TABLE 8. Comparison of η_{Knuth} for various variable- and fixed-length codes with η_2 and η_3 for $(d, k) = (3, 7)$.

$2m$	Const.	η_{Knuth}	η_2	η_3	η_1
$248\,(214+34)$	FL2	0.8426	0.8461	0.8170	0.9395
$300\,(260+40)$	FL1	$0.8499-0.8706$	0.8659	0.8493	0.9491
$352\,(318+34)$	VL2	0.8618	0.8806	0.8626	0.9559
$412\,(378+34)$	VL1	0.8752	0.8946	0.8725	0.9617

the first value is based on maximal cardinality assuming RLL codeword boundaries correspond to run boundaries and the second value without this limitation using [1, Eq. (4.4)]. The actual value is between these two bounds.

For $(d, k) = (1, 3)$, Table 6 shows that η_2 is bigger than η_{Knuth} for various values of $2m$ and different constructions. In the case of η_3, the same applies, except for FL2, where η_{Knuth} is marginally greater. But recall that η_{Knuth} assumes maximal cardinality for the source RLL code, which can be achieved with enumerative coding, but would be more complex than quasi-enumerative coding since interleaved RLL codes are generated at half codeword length. Non-enumerative RLL codes operate at lower efficiency.

In the case of $(d, k) = (1, 9)$ shown in Table 7, a slight majority of the entries have η_2 and η_3 better than η_{Knuth}. Notable are the cases of FL1 and FL2 for larger $2m$ which demonstrate η_{Knuth} larger than η_2 and η_3. The gain is significant. The comparison can be further refined by considering the very efficient rate $8/12$, $(1, 9)$-constrained code developed by Immink [17]. By concatenating the RLL codewords of length 12 to be as close to the source RLL length of 136 for FL1, an efficiency of 0.8405 can be attained, which is still greater than η_2 and η_3. Similarly, in the case of length 338 FL1, an efficiency of 0.9084 can be attained, which is slightly better than η_2 and η_3. However, it should be noted that the $8/12$, $(1, 9)$-constrained encoder consists of 358 states.

For all the entries of Table 8 for $(d, k) = (3, 7)$, η_2 is better than η_{Knuth}, while η_3 is slightly lower in most cases. With non-enumerative RLL codes, the efficiency would be lower than η_3 for the variable-length codes.

For the majority of cases considered, η_2 and η_3 are larger than η_{Knuth}. The significant exception is when the difference between k and d is large.

VI. CONCLUSION

Quasi-enumerative coding is proposed as a means of more efficiently constructing balanced RLL codes. All three proposed quasi-enumerative coding constructions have a significantly lower computational complexity compared to the enumerative coding construction developed by Kurmaev. Constructions 2 and 3, which are based on permutation RLL codes, have a lower computational complexity than Construction 1, but this reduction in computational complexity comes at the cost of reduced code rate efficiency. In comparison to balanced RLL codes based on Knuth-like balancing methods, Constructions 2 and 3 achieve better code rate efficiency in the majority of cases. The three quasi-enumerative coding constructions can cater for different operational requirements in terms of computational complexity and code rate efficiency.

REFERENCES

[1] K. A. S. Immink, *Codes for Mass Data Storage Systems*, 2nd ed. Eindhoven, The Netherlands: Shannon Foundation, 2004.

[2] B. Vasic and E. M. Kurtas, *Coding and Signal Processing for Magnetic Recording Systems*. Boca Raton, FL, USA: CRC Press, 2005.

[3] I. Djordjevic, W. Ryan, and B. Vasic, *Coding for Optical Channels*. New York, NY, USA: Springer, 2010.

[4] K. A. S. Immink, J. H. Weber, and H. C. Ferreira, "Balanced runlength limited codes using Knuth's algorithm," in *Proc. IEEE Int. Symp. Inf. Theory*, St. Petersburg, Russia, Aug. 2011, pp. 317–320.

[5] F. Paluncic, B. T. Maharaj, and H. C. Ferreira, "Variable- and fixed-length balanced runlength-limited codes based on a Knuth-like balancing method," *IEEE Trans. Inf. Theory*, vol. 65, no. 11, pp. 7045–7066, Nov. 2019.

[6] D. Knuth, "Efficient balanced codes," *IEEE Trans. Inf. Theory*, vol. IT-32, no. 1, pp. 51–53, Jan. 1986.

[7] O. F. Kurmaev, "Constant-weight and constant-charge binary run-length limited codes," *IEEE Trans. Inf. Theory*, vol. 57, no. 7, pp. 4497–4515, Jul. 2011.

[8] A. Manada and H. Morita, "On the capacities of balanced codes with run-length constraints," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Aachen, Germany, Jun. 2017, pp. 1391–1395.

[9] F. Paluncic and B. T. J. Maharaj, "Using bivariate generating functions to count the number of balanced runlength-limited words," in *Proc. GLOBECOM IEEE Global Commun. Conf.*, Singapore, Dec. 2017, pp. 1–6.

[10] O. Milenkovic and B. Vasic, "Permutation (d,k) codes: Efficient enumerative coding and phrase length distribution shaping," *IEEE Trans. Inf. Theory*, vol. 46, no. 7, pp. 2671–2675, Nov. 2000.

[11] P. Flajolet and R. Sedgewick, *Analytic Combinatorics*. Cambridge, U.K.: Cambridge Univ. Press, 2009.

[12] M. Abramson, "Restricted combinations and compositions," *Fibonacci Quart.*, vol. 14, no. 5, pp. 439–452, Dec. 1976.

[13] E. Zehavi and J. K. Wolf, "On runlength codes," *IEEE Trans. Inf. Theory*, vol. IT-34, no. 1, pp. 45–54, Jan. 1988.

[14] T. M. Cover, "Enumerative source encoding," *IEEE Trans. Inf. Theory*, vol. IT-19, no. 1, pp. 73–77, Jan. 1973.

[15] A. Hareedy, B. Dabak, and R. Calderbank, "The secret arithmetic of patterns: A general method for designing constrained codes based on lexicographic indexing," *IEEE Trans. Inf. Theory*, vol. 68, no. 9, pp. 5747–5778, Sep. 2022.

[16] S. Datta and S. W. McLaughlin, "An enumerative method for runlength-limited codes: Permutation codes," *IEEE Trans. Inf. Theory*, vol. 45, no. 6, pp. 2199–2204, Sep. 1999.

[17] K. A. S. Immink, "Block-decodable runlength-limited codes via look-ahead technique," *Philips J. Res.*, vol. 46, no. 6, pp. 293–310, 1992.

FILIP PALUNČIĆ (Member, IEEE) was born in Belgrade, Serbia. He received the M.Ing. and D.Ing. degrees from the University of Johannesburg, South Africa, in 2008 and 2012, respectively. He spent four years in industry as a Research and Development Engineer with IDX, a company specializing in industrial communications. From 2016 to 2017, he was a Postdoctoral Research Fellow of broadband wireless multimedia communications with the Sentech Group, Department of Electrical, Electronic and Computer Engineering, University of Pretoria, where he is currently a Faculty Member of the Department of Electrical, Electronic and Computer Engineering. His research interests include coding techniques (in particular error control coding and constrained coding), information theory, cognitive radio networks, and wireless communications.

B. T. (SUNIL) MAHARAJ (Senior Member, IEEE) received the Ph.D. degree in engineering in the areas of wireless communications from the University of Pretoria. He is currently a Full Professor and holds the research position with the Sentech Chair in Broadband Wireless Multimedia Communications, Department of Electrical, Electronic and Computer Engineering, University of Pretoria. His research interests include OFDM-MIMO, massive MIMO systems, cognitive radio resource allocation, and 5G cognitive radio sensor networks.