Variable- and Fixed-Length Balanced Runlength-Limited Codes Based on a Knuth-Like Balancing Method

Filip Palunčić, *Member, IEEE*, B. T. Maharaj, *Senior Member, IEEE*, and Hendrik C. Ferreira†, *Senior Member, IEEE*

***Abstract*—A novel Knuth-like balancing method for runlength-limited words is presented, which forms the basis of new variable- and fixed-length balanced runlength-limited codes that improve on the code rate as compared to balanced runlength-limited codes based on Knuth's original balancing procedure developed by Immink *et al.* While Knuth's original balancing procedure, as incorporated by Immink *et al.*, requires the inversion of each bit one at a time, our balancing procedure only inverts the runs as a whole one at a time. The advantage of this approach is that the number of possible inversion points, which needs to be encoded by a redundancy-contributing prefix/suffix, is reduced, thereby allowing a better code rate to be achieved. Furthermore, this balancing method also allows for runlength violating markers which improve, in a number of respects, on the optimal such markers based on Knuth's original balancing method.**

***Index Terms*—Knuth-like balancing method, balanced codes, runlength-limited codes, random walk.**

I. Introduction

Constrained codes endow random data with particular desirable properties which have important applications, for example, in magnetic and optical storage media [1]–[3]. Runlength-limited (RLL) and DC-free codes are examples of constrained codes. RLL codes limit the length of a run, which is a maximal sub-word consisting of like symbols, to lie between inclusive lower and upper bounds. We will denote the inclusive lower bound by d' and the inclusive upper bound by k' and refer to a RLL code defined by these two parameters as a (d', k')-RLL code. The lower bound d' is used to limit inter-symbol interference in bandwidth-limited channels, whereas the upper bound k' facilitates maintenance of synchronization. On the other hand, DC-free codes possess the virtue that the power spectral density (PSD) of the analogue signal representation of binary words is zero at zero (DC) frequency and close to zero for frequencies near zero frequency.

Given the importance of these two types of constrained codes in their own right, it is natural that DC-free RLL codes, which combine both these properties concurrently, occupy a prominent position in the family of constrained codes. This is indicated by an entire chapter dedicated to DC-free RLL codes in Immink's book on constrained codes [1, Ch. 11]. Some of the various approaches to constructing DC-free RLL codes include the use of merging sequences of different weight parities for RLL block codes [4], guided scrambling [5] and enumerative coding [6]. Enumerative coding is attractive due to its high code rate efficiency, but as emphasized by Immink *et al.* [7], they suffer from prohibitively large memory requirements for growing codeword lengths and massive error propagation.

A particular manifestation of DC-free codes are balanced codes which consist of codewords containing an equal number of the two symbols from some binary alphabet. Part of the attractiveness of balanced codes stems from the existence of a simple encoding/decoding method proposed by Knuth [8]. This method consists of sequentially inverting (complementing) one bit at a time for a binary sequence. Knuth proved that there exists at least one index such that the resulting sequence is balanced. This index, referred to as a *balancing index*, which is required at the decoder to extract the original sequence, is encoded by means of a balanced prefix/suffix which is concatenated with the partially complemented balanced sequence. This method provides an efficient means of encoding and decoding very large word lengths and scales well with growing word length. This is the prime motivation why Immink *et al.* [7] utilized Knuth's balancing method in their construction of balanced RLL codes. This approach yields very efficient codes in terms of code rate. Various refinements and improvements to Knuth's method for balanced codes can be found in [9]–[13].

Kurmaev [14] presented an enumerative coding procedure for charge-constrained RLL codes (of which balanced RLL codes are a special case). Although these codes have a high code rate efficiency, they are not suitable for long codeword lengths due to mounting computational complexity/memory requirements for growing codeword lengths.

Although the capacity of DC-free RLL codes has been known for some time [15], it was only recently shown that the capacity of balanced RLL codes is in fact equal to the capacity of the corresponding RLL codes [16]. This result means that the additional balancing constraint does not incur a rate loss asymptotically. This is consistent with the observation that balanced codes have a capacity of 1 [16, §II]. However,

F. Palunčić and B. T. Maharaj are with the Department of Electrical, Electronic and Computer Engineering, University of Pretoria, Pretoria 0002, South Africa (e-mail: filip.paluncic@up.ac.za, sunil.maharaj@up.ac.za).

H. C. Ferreira† was with the Department of Electrical and Electronic Engineering Science, University of Johannesburg, South Africa (e-mail: hcferreira@uj.ac.za). †Deceased on 30 November 2018.

This work was supported by the National Research Foundation (NRF) and SENTECH Chair in Broadband Wireless Multimedia Communication.

as indicated in [17, §I], balanced codes are DC-free only for finite word lengths, with the implication that capacity-reaching balanced RLL codes are not DC-free. This is what motivated the use of generating functions to count the number of balanced RLL words of a specific, finite length as an alternative to capacity in evaluating the performance of balanced RLL codes in terms of code rate [17].

In this paper, we propose a novel Knuth-like balancing method which allows the construction of balanced RLL codes which improve on the code rates of the corresponding balanced RLL code constructions by Immink *et al.* [7]. Our balancing procedure sequentially inverts runs as a whole one at a time, unlike the original method by Knuth, incorporated by Immink *et al.* [7], which inverts each bit one at a time. The advantage of this approach is that the number of possible balancing positions, which has to be encoded by a redundancy-contributing prefix/suffix, is reduced, leading to a lower redundancy. Based on this alternative balancing approach, variable- and fixed-length balanced RLL codes are constructed. The proposed variable-length balanced RLL codes improve on the code rate of the codes by Immink *et al.* [7] for all finite d' and k' (with the exception of $d' = 1$ for smaller source word lengths), while the proposed fixed-length codes have an improved code rate with the exception when $d' = 1$. The variable-length codes have the best code rate performance and are also attractive as they do not have certain restrictions on d' and k' inherent in the fixed-length codes of this paper and of Immink *et al.* [7]. A disadvantage of our codes, both variable- and fixed-length, as compared to that of Immink *et al.* [7], is that they are not suited for the case where $k' = \infty$.

Codes using markers with deliberate runlength violations to signify the balancing index obtained using Knuth's original method have been proposed for balanced RLL codes [18]. The length of such markers, which equal the total redundancy of the code, is fixed and independent of source word length. Thus, these codes, for a sufficiently large codeword length, have better code rates than the codes presented by Immink *et al.* [7] and the codes presented in this paper. Weber *et al.* [19] studied optimal runlength-violating markers based on Knuth's original balancing method in terms of marker length and cumulative violation, determining the minimum violation of markers with the smallest possible length and the smallest length of unit violation markers. Based on the novel Knuth-like balancing method, we present a variable-length runlength-violating marker that improves, in a number of respects, on the optimal markers based on Knuth's original balancing method presented in [19]. Therefore, the novel Knuth-like balancing method presented in this paper improves on both existing balanced RLL codes with stringent runlength constraints (i.e. where runlength violations are not tolerable) and runlength-violating markers based on Knuth's original balancing method.

This paper is organized as follows. Section II introduces the notation and gives a succinct overview of pertinent concepts. A modification of Knuth's balancing method for RLL words and relevant proofs are presented in Section III. The various generating functions pertinent to the determination of the code's redundancy are derived in Section IV, while the encoding and decoding algorithms for both the variable- and fixed-length versions of the code are formalized in Section V. Section VI contains a performance comparison in terms of code rate between the codes proposed in this paper and those from [7]. Concluding remarks are contained in Section VII.

II. Notation and Background

This section serves to introduce the notation employed in this paper and provides an overview of the pertinent literature. Apart from defining balanced codes and describing Knuth's balancing method, it also introduces the various runlength-limited sequence representations as applicable to this paper and overviews existing balanced RLL codes based on Knuth's balancing method.

A. Balanced Codes

For a bipolar word $\boldsymbol{x} = (x_1, x_2, \ldots, x_m) \in \{-1, +1\}^m$ of length m, define $\sigma(\boldsymbol{x})$ as

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

A word $\boldsymbol{x}$, where m is even, is said to be *balanced* if, and only if, $\sigma(\boldsymbol{x}) = 0$, i.e., it consists of an equal number of the symbols '-1' and '$+1$'. A code is said to be *balanced* if each word in the code is balanced.

B. Knuth's Balancing Method

For a bipolar word $\boldsymbol{x} = (x_1, x_2, \ldots, x_m) \in \{-1, +1\}^m$, let $\boldsymbol{x}^{[i]}$ denote $\boldsymbol{x}$ with the first i symbols inverted, i.e., $\boldsymbol{x}^{[i]} \triangleq (-x_1, -x_2, \ldots, -x_i, x_{i+1}, x_{i+2}, \ldots, x_m)$. It is clear that $\boldsymbol{x}^{[0]} = \boldsymbol{x}$ and $\boldsymbol{x}^{[m]} = -\boldsymbol{x}$. Then, Knuth's method iterates over i, $0 \leq i \leq m$, starting at $i = 0$, until the first $\sigma(\boldsymbol{x}^{[i]}) = 0$ is found. Any i such that $\boldsymbol{x}^{[i]}$ is balanced is referred to as a *balancing index*. It is guaranteed that for any word $\boldsymbol{x}$ there exists at least one balancing index i [8, pp. 51-52]. This can be demonstrated by considering the random walk $(i, \sigma(\boldsymbol{x}^{[i]}))$, $0 \leq i \leq m$, of $\boldsymbol{x}$. By noting that for even m, $\sigma(\boldsymbol{x})$ is also even, $\sigma(\boldsymbol{x}^{[m]}) = -\sigma(\boldsymbol{x}^{[0]} = \boldsymbol{x})$ and $\sigma(\boldsymbol{x}^{[i]}) - \sigma(\boldsymbol{x}^{[i-1]}) = \pm 2$, it follows that random walk of $\boldsymbol{x}$ has to pass through $(i, 0)$ for at least one i, $0 \leq i \leq m$.

C. Runlength-Limited Sequence Representations

A *run* is formally defined as the maximal sub-sequence consisting of like symbols. As already indicated, a (d', k')-RLL code is a code where each codeword consists of runlengths of at least d' and at most k'. A code is said to be a (d, k)-constrained code if the number of zeros between consecutive ones is at least d and at most k for each codeword. These are alternative representations of runlength-limited sequences as can be shown by the mapping $\boldsymbol{f} : \boldsymbol{y} = (y_1, y_2, \ldots, y_m) \in \{0, 1\}^m \rightarrow \boldsymbol{z} = (z_1, z_2, \ldots, z_m) \in \{-1, +1\}^m$. We assume that $\boldsymbol{y}$ represents a (d, k)-constrained sequence of length m and that $\boldsymbol{z}$ represents the corresponding (d', k')-RLL sequence over the bipolar alphabet $\{-1, +1\}$ of the same length. Then, for $\boldsymbol{z} = \boldsymbol{f}(\boldsymbol{y})$

$$\hat{z}_i = \hat{z}_{i-1} \oplus y_i,$$
$$z_i = 2\hat{z}_i - 1,$$

where $1 \leq i \leq m$, $\hat{z}_0 = 1$ and $\oplus$ denotes modulo 2 addition. For the inverse operation $\boldsymbol{y} = \boldsymbol{f}^{-1}(\boldsymbol{z})$, $y_i = (-z_{i-1}z_i + 1)/2$. Note that a one in $\boldsymbol{y}$ corresponds to a transition in $\boldsymbol{z}$, i.e., $y_i = 1$ implies that $z_i = -z_{i-1}$, while a zero in $\boldsymbol{y}$ corresponds to a non-transition in $\boldsymbol{z}$, i.e., $y_i = 0$ implies that $z_i = z_{i-1}$. Hence, it is easy to see that a (d, k)-constrained code corresponds to a $(d + 1, k + 1)$-RLL code, i.e., $d' = d + 1$ and $k' = k + 1$.

Now we introduce another representation for RLL sequences. First, let $\mathcal{L}_{(d',k')}$ denote the set of permissible runlengths, i.e.,

$$\mathcal{L}_{(d',k')} \triangleq \{d', d' + 1, \ldots, k'\}.$$

Also, $a\mathcal{L}_{(d',k')} \triangleq \{al : l \in \mathcal{L}_{(d',k')}\}$ and $a + \mathcal{L}_{(d',k')} = \mathcal{L}_{(d',k')} + a \triangleq \{a + l : l \in \mathcal{L}_{(d',k')}\}$. Assume that $\boldsymbol{z} = (z_1, z_2, \ldots, z_m) \in \{-1, +1\}^m$ is a (d', k')-RLL word where the first and last runs also satisfy the runlength constraint. Let $n(\boldsymbol{z})$ denote the number of runs in $\boldsymbol{z}$ and let $\boldsymbol{\rho}(\boldsymbol{z}) = (\rho_1, \rho_2, \ldots, \rho_{n(\boldsymbol{z})})$ be a sequence which denotes the runlengths in $\boldsymbol{z}$, i.e., $\rho_i \in \mathcal{L}_{(d',k')}$ is the length of the ith run in $\boldsymbol{z}$. Note that

$$m = \sum_{i=1}^{n(\boldsymbol{z})} \rho_i.$$

We also introduce a bipolar sequence $\boldsymbol{\beta}(\boldsymbol{z}) = (\beta_1, \beta_2, \ldots, \beta_{n(\boldsymbol{z})}) \in \{-1, +1\}^{n(\boldsymbol{z})}$ to represent the polarity of the runs, where $\beta_1 \triangleq z_1$. In fact, $\boldsymbol{\beta}(\boldsymbol{z})$ is totally defined by β_1 as $\beta_i = -\beta_{i-1}$, $2 \leq i \leq n(\boldsymbol{z})$. $\boldsymbol{\beta}(\boldsymbol{z})$ is defined in this manner because the polarity alternates between adjacent runs. We will refer to $\boldsymbol{\beta}(\boldsymbol{z})$ as a *polarity word*. Then, it is clear that

$$\sigma(\boldsymbol{z}) = \sigma(\boldsymbol{\beta}(\boldsymbol{z}) \odot \boldsymbol{\rho}(\boldsymbol{z})),$$

where $\odot$ represents the Hadamard product, i.e., $\boldsymbol{\beta}(\boldsymbol{z}) \odot \boldsymbol{\rho}(\boldsymbol{z}) = (\beta_1\rho_1, \beta_2\rho_2, \ldots, \beta_{n(\boldsymbol{z})}\rho_{n(\boldsymbol{z})})$, and

$$\sigma(\boldsymbol{\beta}(\boldsymbol{z}) \odot \boldsymbol{\rho}(\boldsymbol{z})) = \sum_{i=1}^{n(\boldsymbol{z})} \beta_i\rho_i.$$

For example, with $- \equiv -1$ and $+ \equiv +1$, if $\boldsymbol{z} = (--+++++---)$, then $n(\boldsymbol{z}) = 3$, $\boldsymbol{\rho}(\boldsymbol{z}) = (2, 4, 3)$ and $\boldsymbol{\beta}(\boldsymbol{z}) = (-+-)$, and hence $\sigma(\boldsymbol{z}) = \sigma(\boldsymbol{\beta}(\boldsymbol{z}) \odot \boldsymbol{\rho}(\boldsymbol{z})) = -2 + 4 - 3 = -1$.

In the sequel, for notational convenience, we are going to drop the argument $\boldsymbol{z}$ from n, $\boldsymbol{\rho}$ and $\boldsymbol{\beta}$, and simply represent the equivalence between $\boldsymbol{z}$ and $\boldsymbol{\rho}$, $\boldsymbol{\beta}$ (recall that $\boldsymbol{\beta}$ is fully characterized by β_1) as

$$\boldsymbol{z} \Leftrightarrow \boldsymbol{\rho}^{(\beta_1)}.$$

This equivalence can also be characterized by

$$\boldsymbol{z} = (\beta_1^{\rho_1}, \beta_2^{\rho_2}, \ldots, \beta_n^{\rho_n}),$$

where for some symbol s, $s^i \triangleq (s, s, \ldots, s)$ is a word of length i consisting only of the repeating symbol s. Note, furthermore, that the inversion (complementation) of $\boldsymbol{z}$ is equivalent to the inversion (complementation) of the polarity word $\boldsymbol{\beta}$, i.e., if $\boldsymbol{z} \Leftrightarrow \boldsymbol{\rho}^{(\beta_1)}$, then $-\boldsymbol{z} \Leftrightarrow \boldsymbol{\rho}^{(-\beta_1)}$.

D. Balanced Runlength-Limited Codes based on Knuth's Balancing Method

Immink *et al.* [7] exploited Knuth's balancing method to construct efficient balanced RLL codes. To a (d, k)-constrained word $\boldsymbol{y} = (y_1, y_2, \ldots, y_m) \in \{0, 1\}^m$ produced by some prior art method, Knuth's balancing method is applied to $\boldsymbol{f}(\boldsymbol{y})$ until a balanced word $\boldsymbol{f}(\boldsymbol{y})^{[i]}$ is obtained. However, since the balancing index i can occur within a run, the inversion applied during Knuth's balancing method can lead to runlength constraint violations, requiring the use of a judiciously chosen interfix, which is inserted immediately after the balancing index, to ensure the maintenance of the runlength constraints. In the process of preserving the runlength constraints, the interfix may introduce a limited unbalance to the balanced word produced by Knuth's procedure. In addition to this fixed-length interfix, a fixed-length suffix is also needed to encode the balancing index i. Apart from encoding the balancing index, the suffix has an unbalance equal in magnitude, but opposite in polarity, to that introduced by the interfix, resulting in an overall balanced word. The redundancy of the code is the sum of the lengths of the interfix and suffix.

There exists some bijective mapping from each possible balancing index i to a distinct suffix word which possesses certain desirable characteristics (these characteristics are elaborated on in Sec. IV). The decoder, which is cognizant of this bijective mapping, first extracts the suffix, from which it can deduce the balancing index. Once the balancing index is known, the fixed-length interfix can be removed and inversion of Knuth's procedure undone, resulting in the original (d, k)-constrained word.

Immink *et al.* [7] presented four constructions, each considering different cases based on values of d and k. Constructions 1 and 2 are for the case where there is no upper constraint (i.e. $k = \infty$), Construction 3 is for the general case where d ($d > 0$) and k are finite with the restriction that $k \geq 2d$, and Construction 4 is for the case where there is no lower constraint (i.e. $d = 0$) and k is finite. Construction 1's interfix length is $2(d + 1)$, Construction 2 and 3's is $d + 1$, while Construction 4's is 1. Construction 1 is the only construction where the modified $\boldsymbol{y}$ (through Knuth's balancing method) plus interfix and the suffix are both balanced, while for the other constructions the unbalanced of the modified $\boldsymbol{y}$ plus interfix is compensated for by the unbalance of the suffix.

A limitation of the codes presented in [7] is that the balanced RLL codewords produced by the code cannot be freely cascaded. This is because, although the constructions ensure that the runlength constraints are maintained internally within the codeword, there is no mechanism which ensures that the runlength constraints are not violated at codeword boundaries. This limitation of the codes from [7] will be further elaborated on in Sec. VI.

An alternative method to indicating the balancing index was proposed by Ferreira *et al.* [18]. Immediately after the balancing index is inserted a fixed-length marker (conceptually similar to the interfix from [7]) whose first run deliberately violates the k constraint. The decoder can then use this

violation to identify the balancing index. This deliberate and controlled violation is motivated by advances in phase-lock loops and the associated greater tolerance for occasional violations. While the marker has to be balanced and maintain the runlength constraints at the balancing index which affect its length, this approach has the virtue that the marker length is independent of codeword length. Therefore, for sufficiently large codeword lengths, they have a better code rate than the balanced RLL codes from [7]. Subsequently, Weber *et al.* [19] studied such markers and formalized their desirable properties. They demonstrated that the shortest possible length of such markers is $2k+2$, which have a minimum possible violation of $k+1$ (here violation is defined as the cumulative violation size of all the marker violations, where the size of a violation is the amount that a runlength exceeds k), and that for the smallest possible violation of 1, the minimum marker length is $2k+4d+6$. They also gave examples of optimal fixed- and variable-length markers which achieve these bounds.

In the next section, we present a novel Knuth-like balancing method for RLL words, which forms the basis of balanced RLL codes which improve on the balanced RLL codes from [7] and markers which improve on the optimal markers based on Knuth's original balancing method presented in [19].

III. Knuth-Like Balancing Method for RLL Words

The novel Knuth-like balancing method for RLL words is based on the idea of inverting (or complementing) complete runs as a whole rather than inverting bits individually. An advantage of this approach is the reduction in the number of possible balancing indices. Furthermore, since the inversions occur at run boundaries, the interfix can be a single run, simplifying the process of maintaining the runlength constraint at the balancing index.

The balancing method can be described as follows. Given $\boldsymbol{z} \Leftrightarrow \boldsymbol{\rho}^{(\beta_1)}$, we wish to balance $\boldsymbol{z}$, which is a (d',k')-RLL word. For $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_n)$, let $\boldsymbol{\beta}^{[i]}$ denote $\boldsymbol{\beta}$ with the first i symbols inverted, i.e., $\boldsymbol{\beta}^{[i]} \triangleq (-\beta_1, -\beta_2, \ldots, -\beta_i, \beta_{i+1}, \ldots, \beta_n)$. Note that since $\boldsymbol{\beta}^{[0]} = \boldsymbol{\beta}$ and $\boldsymbol{\beta}^{[n]} = -\boldsymbol{\beta}$, $\sigma(\boldsymbol{\beta}^{[n]} \odot \boldsymbol{\rho}) = -\sigma(\boldsymbol{\beta}^{[0]} \odot \boldsymbol{\rho})$. The balancing procedure consists of inverting the symbols in $\boldsymbol{\beta}$ one at a time, i.e., iterating over i, $0 \le i \le n$. Associated with this balancing procedure, we introduce the random walk

$$\Big\{(i, \sigma(\boldsymbol{\beta}^{[i]} \odot \boldsymbol{\rho})) : i = 0, 1, \ldots, n\Big\}.$$

We will refer to such a random walk as a RLL random walk. For notational convenience, we are going to denote $\sigma(\boldsymbol{\beta}^{[i]} \odot \boldsymbol{\rho})$ simply as σ_i.

Example 1: Consider the $(d'=2, k'=4)$-RLL word

$$\boldsymbol{z} = (----++--++++--).$$

Then for $\boldsymbol{z} \Leftrightarrow \boldsymbol{\rho}^{(-)}$, $n=5$, $\boldsymbol{\rho} = (4,3,2,4,3)$ and $\boldsymbol{\beta} = (-+-+-)$. Then the random walk (i, σ_i), $0 \le i \le 5$, is

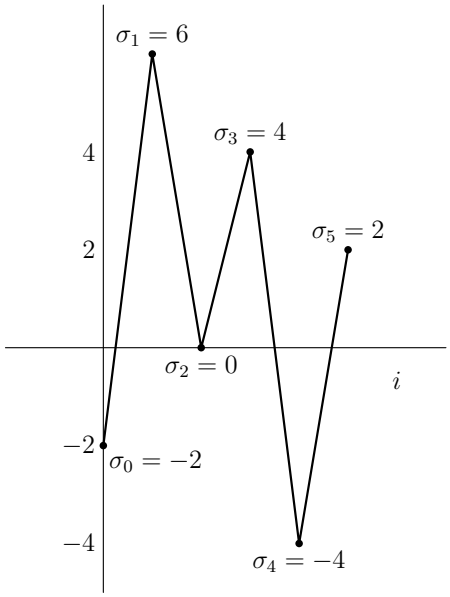

Fig. 1. The RLL random walk (i, σ_i), $0 \le i \le 5$, of $\boldsymbol{z} \Leftrightarrow \boldsymbol{\rho}^{(-)}$, where $\boldsymbol{\rho} = (4,3,2,4,3)$.

$(0,-2) \to (1,6) \to (2,0) \to (3,4) \to (4,-4) \to (5,2)$ since

$i = 0 : \boldsymbol{\beta}^{[0]} = (-+-+-), \sigma_0 = -4+3-2+4-3 = -2$

$i = 1 : \boldsymbol{\beta}^{[1]} = (\underline{+}+-+-), \sigma_1 = 6$

$i = 2 : \boldsymbol{\beta}^{[2]} = (\underline{+-}-+-), \sigma_2 = 0$

$i = 3 : \boldsymbol{\beta}^{[3]} = (\underline{+-+}+-), \sigma_3 = 4$

$i = 4 : \boldsymbol{\beta}^{[4]} = (\underline{+-+-}-), \sigma_4 = -4$

$i = 5 : \boldsymbol{\beta}^{[5]} = (\underline{+-+-+}), \sigma_5 = 2.$

Underlining signifies the inverted portions of $\boldsymbol{\beta}^{[i]}$. The random walk (i, σ_i) is shown in Fig. 1. □

Now we introduce and define useful terminology related to the random walk (i, σ_i). First, we define the meaning of an *ascent* and *descent* at σ_i, $1 \le i \le n$:

- **Ascent** at σ_i: the transition $\sigma_{i-1} \to \sigma_i$ is such that $\sigma_i > \sigma_{i-1}$. This occurs when $\beta_i = -1$.
- **Descent** at σ_i: the transition $\sigma_{i-1} \to \sigma_i$ is such that $\sigma_i < \sigma_{i-1}$. This occurs when $\beta_i = +1$.

From Fig. 1, it can be seen that σ_1, σ_3 and σ_5 are ascents, while σ_2 and σ_4 are descents. Then a *peak* and *valley* at σ_i, $0 \le i \le n$, can be defined as:

- **Peak** at σ_i:
  1) $1 \le i \le n-1$: σ_i is an ascent and σ_{i+1} is a descent.
  2) $i = 0$: σ_1 is a descent.
  3) $i = n$: σ_n is an ascent.
- **Valley** at σ_i:
  1) $1 \le i \le n-1$: σ_i is a descent and σ_{i+1} is an ascent.
  2) $i = 0$: σ_1 is an ascent.
  3) $i = n$: σ_n is a descent.

From Fig. 1, it can be seen that σ_1, σ_3 and σ_5 are peaks, while σ_0, σ_2 and σ_4 are valleys. Since $\boldsymbol{\beta}$ consists of alternating '-1' and '$+1$' symbols, the corresponding random walk consists of alternating ascents and descents, which in turn implies that the random walk consists of alternating peaks and valleys.

The range of the random walk (i, σ_i) is partitioned into two broad regions:

- **Inner region**: $-k' \leq \sigma_i \leq k'$.
- **Outer region**: $\sigma_i < -k'$ and $\sigma_i > k'$.

Furthermore, the inner region is partitioned into three sub-regions:

- **Positive primary region**: $d' \leq \sigma_i \leq k'$.
- **Secondary region**: $-d' < \sigma_i < d'$.
- **Negative primary region**: $-k' \leq \sigma_i \leq -d'$.

The *primary region* is the union of the positive and negative primary regions. This terminology is summarized in Fig. 2.

Note that inverting a symbol in $\boldsymbol{\beta}$ changes the polarity of the corresponding run in $\boldsymbol{\rho}$, thereby producing the inversion of the entire run. Also, since $\boldsymbol{\beta}$ consists of alternating '-1's and '$+1$'s, at the inversion index i, $-\beta_i = \beta_{i+1}$, meaning that the ith and $(i+1)$th runs are merged since they have the same polarity, leading to a potential runlength violation. To ensure the preservation of the runlength constraints, an interfix consisting of a single run of length $d' \leq \rho' \leq k'$ is inserted into $\boldsymbol{\rho}$ at index $i+1$, $0 \leq i \leq n$, giving

$$\boldsymbol{\rho'} \triangleq (\rho_1, \rho_2, \ldots, \rho_i, \rho', \rho_{i+1}, \ldots, \rho_n). \tag{1}$$

Note that ρ' can be prepended to $\boldsymbol{\rho}$ ($i = 0$: insertion at index 1) or appended to $\boldsymbol{\rho}$ ($i = n$: insertion at "virtual" index $n+1$). Similarly,

$$\beta' \triangleq -(-\beta_i) = \beta_i = -\beta_{i+1} \tag{2}$$

is inserted into $\boldsymbol{\beta}^{[i]}$ at index $i+1$, $0 \leq i \leq n$, giving

$$\boldsymbol{\beta'}^{[i]} = (-\beta_1, -\beta_2, \ldots, -\beta_i, \beta', \beta_{i+1}, \ldots, \beta_n). \tag{3}$$

If $i = 0$, then $\beta' \triangleq -\beta_1$, and if $i = n$, then $\beta' \triangleq -(-\beta_n) = \beta_n$. It is clear that $\boldsymbol{\beta'}^{[i]}$, like $\boldsymbol{\beta}$, consists of alternating '-1' and '$+1$' symbols. Therefore, it follows that $\boldsymbol{z'} \Leftrightarrow \boldsymbol{\rho'}^{(\beta'_1)}$ is a (d', k')-RLL word, where β'_1 is the first symbol in $\boldsymbol{\beta'}^{[i]}$ ($\beta'_1 = \beta'$ if $i = 0$ and $\beta'_1 = -\beta_1$ if $i > 0$).

In order to obtain a balanced RLL word $\boldsymbol{z'} \Leftrightarrow \boldsymbol{\rho'}^{(\beta'_1)}$ ($\sigma(\boldsymbol{z'}) = \sigma(\boldsymbol{\beta'}^{[i]} \odot \boldsymbol{\rho'}) = 0$), at least one index i, $0 \leq i \leq n$, needs to exist such that $d' \leq \sigma(\boldsymbol{\beta}^{[i]} \odot \boldsymbol{\rho}) \leq k'$ if $\beta' = -1$ or $-k' \leq \sigma(\boldsymbol{\beta}^{[i]} \odot \boldsymbol{\rho}) \leq -d'$ if $\beta' = +1$. Then by selecting $\rho' = |\sigma(\boldsymbol{\beta}^{[i]} \odot \boldsymbol{\rho})|$, $\boldsymbol{z'} \Leftrightarrow \boldsymbol{\rho'}^{(\beta'_1)}$ will be balanced. The first case where $d' \leq \sigma(\boldsymbol{\beta}^{[i]} \odot \boldsymbol{\rho}) \leq k'$ if $\beta' = -1$ corresponds to a peak at σ_i in the positive primary region, as can be shown by noting the following:

- According to (2), $\beta' = -1$ requires that $\beta_i = \beta' = -1$ and $\beta_{i+1} = -\beta' = +1$. $\beta_i = -1$ produces an ascent at σ_i (if $i \neq 0$), while $\beta_{i+1} = +1$ produces a descent at σ_{i+1} (if $i \neq n$), meaning that σ_i is a peak.
- Since $d' \leq \sigma_i \leq k'$, the peak occurs in the positive primary region.

Similarly, the second case where $-k' \leq \sigma(\boldsymbol{\beta}^{[i]} \odot \boldsymbol{\rho}) \leq -d'$ if $\beta' = +1$ corresponds to a valley at σ_i in the negative primary region, as can be shown by noting the following:

- According to (2), $\beta' = +1$ requires that $\beta_i = \beta' = +1$ and $\beta_{i+1} = -\beta' = -1$. $\beta_i = +1$ produces a descent at σ_i (if $i \neq 0$), while $\beta_{i+1} = -1$ produces an ascent at σ_{i+1} (if $i \neq n$), meaning that σ_i is a valley.
- Since $-k' \leq \sigma_i \leq -d'$, the valley occurs in the negative primary region.

Therefore, it follows that if $\boldsymbol{z} \Leftrightarrow \boldsymbol{\rho}^{(\beta_1)}$ is a (d', k')-RLL word, then $\boldsymbol{z'} \Leftrightarrow \boldsymbol{\rho'}^{(\beta'_1)}$ can only be balanced for some i, $0 \leq i \leq n$, if the random walk (i, σ_i) contains at least one index i such that σ_i is a peak in the positive primary region or a valley in the negative primary region.

In the context of the random walk (i, σ_i) for some (d', k')-RLL word $\boldsymbol{z} \Leftrightarrow \boldsymbol{\rho}^{(\beta_1)}$, the index i, $0 \leq i \leq n$, is said to be a *balancing index* if

- σ_i is a peak in the positive primary region, i.e., $\sigma(\boldsymbol{\beta}^{[i]} \odot \boldsymbol{\rho}) \in \mathcal{L}_{(d',k')}$ and $\beta_i = -1$ for $i > 0$ or $\beta_1 = +1$ for $i = 0$, or
- σ_i is a valley in the negative primary region, i.e., $\sigma(\boldsymbol{\beta}^{[i]} \odot \boldsymbol{\rho}) \in -\mathcal{L}_{(d',k')}$ and $\beta_i = +1$ for $i > 0$ or $\beta_1 = -1$ for $i = 0$.

From Fig. 1 it can be seen that σ_0 and σ_4 are valleys in the negative primary region and that σ_3 and σ_5 are peaks in the positive primary region. Therefore, $i = 0, 3, 4, 5$ are balancing indices. The balanced RLL words $\boldsymbol{z'} \Leftrightarrow \boldsymbol{\rho'}^{(\beta'_1)}$ corresponding to these balancing indices are:

- $i = 0$: $\boldsymbol{\rho'}^{(+)}$, $\boldsymbol{\rho'} = (2, 4, 3, 2, 4, 3)$,
- $i = 3$: $\boldsymbol{\rho'}^{(+)}$, $\boldsymbol{\rho'} = (4, 3, 2, 4, 4, 3)$,
- $i = 4$: $\boldsymbol{\rho'}^{(+)}$, $\boldsymbol{\rho'} = (4, 3, 2, 4, 4, 3)$,
- $i = 5$: $\boldsymbol{\rho'}^{(+)}$, $\boldsymbol{\rho'} = (4, 3, 2, 4, 3, 2)$.

As is proven in Theorem 1, any RLL word is guaranteed to have at least one balancing index provided that n, the number of runs, is odd. Before presenting Theorem 1, we give the following useful lemma.

Lemma 1: For a RLL random walk (i, σ_i) corresponding to some (d', k')-RLL word $\boldsymbol{z} \Leftrightarrow \boldsymbol{\rho}^{(\beta_1)}$, if there exist indices h and j ($0 \leq h < h+2 < j \leq n$) such that $\sigma_h > k'$ ($\sigma_h < -k'$) and $\sigma_j < -k'$ ($\sigma_j > k'$), then there exists at least one balancing index i, $h < i < j$.

Proof: For i to be a balancing index, σ_i has to be a peak in the positive primary region or a valley in the negative primary region.

Since $|\sigma_l - \sigma_{l-1}| \in 2\mathcal{L}_{(d',k')}$, $1 \leq l \leq n$, it is impossible to traverse the inner region in a single random walk step, i.e., for some l, $h+1 \leq l \leq j$, if $\sigma_{l-1} > k'$ ($\sigma_{l-1} < -k'$) and $\sigma_l < -k'$ ($\sigma_l > k'$), it follows that $|\sigma_l - \sigma_{l-1}| > 2k'$. However, $\max|\sigma_l - \sigma_{l-1}| = 2k'$. Therefore, the inner region can be traversed in a minimum of three random walk steps (recall that the random walk (i, σ_i) consists of alternating peaks and valleys), hence $h + 2 < j$.

First consider the case $\sigma_h > k'$ and $\sigma_j < -k'$. Let $h' > h$ be the smallest index such $\sigma_{h'-1} > k'$ and $\sigma_{h'} \leq k'$, i.e., h' is the first index after h that the random walk enters the inner region. Then, there are three possible scenarios, which are depicted in Fig. 3, Fig. 4 and Fig. 5. In the first scenario, depicted in Fig. 3, the random walk enters the inner region where $-d' < \sigma_{h'} \leq k'$. Note that $\sigma_{h'}$ is a valley that is outside the negative primary region. The next step in the random walk then exits the inner region ($\sigma_{h'+1} > k'$), and so we return to the initial conditions. As result, we are going to ignore this scenario by defining h'' to be the smallest index greater than h ($h'' > h$) such that $\sigma_{h''}$ is in the positive primary or

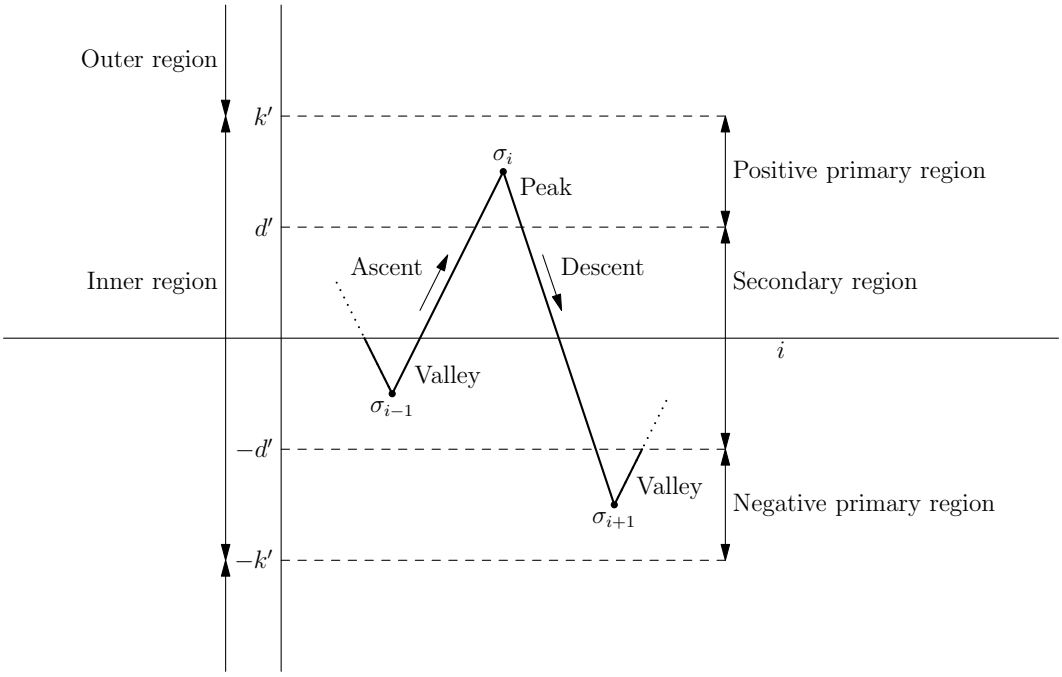

Fig. 2. A generic RLL random walk demonstrating the associated terminology.

secondary region and $\sigma_{h''+1}$ is in the positive primary region, i.e., $\sigma_{h''-1} > k'$, $-d' < \sigma_{h''} \leq k' - 2d'$ and $d' < \sigma_{h''+1} \leq k'$, or $\sigma_{h''}$ is in the negative primary region, i.e., $\sigma_{h''-1} > k'$ and $-k' \leq \sigma_{h''} \leq -d'$. These two cases are depicted in Fig. 4 and Fig. 5, respectively, with a change of h' to h''. Then

1) $-d' < \sigma_{h''} \leq k' - 2d'$: Since $\sigma_{h''}$ is a descent, $\sigma_{h''+1}$ must be an ascent. Furthermore, note that since $\min|\sigma_{h''+1} - \sigma_{h''}| = 2d'$, $\sigma_{h''+1} > d'$. Then, by the definition of h'', $d' < \sigma_{h''+1} \leq k'$, and so $\sigma_{h''+1}$ is a peak in the positive primary region and so $i = h'' + 1$ is a balancing index. Also, note that $i = h'' + 1 < j$ $(h'' - 1 + 2 < j)$.
2) $-k' \leq \sigma_{h''} \leq -d'$: Since $\sigma_{h'}$ is a descent, $\sigma_{h''+1}$ must be an ascent. Therefore, $\sigma_{h''}$ is a valley in the negative primary region and hence $i = h''$ is a balancing index. Also, note that $i = h'' < j$ $(h'' - 1 + 2 < j)$.

By symmetry, the case $\sigma_h < -k'$ and $\sigma_j > k'$ is equivalent to the case $\sigma_h > k'$ and $\sigma_j < -k'$. Therefore, there exists at least one balancing index i, $h < i < j$. ■

Theorem 1: For a RLL random walk (i, σ_i) corresponding to any (d', k')-RLL word $\boldsymbol{z} \Leftrightarrow \boldsymbol{\rho}^{(\beta_1)}$, there exists at least one balancing index i, $0 \leq i \leq n$, if n is odd.

Proof: Firstly, note that $\sigma_n = -\sigma_0$ because $\boldsymbol{\beta}^{[n]} = -\boldsymbol{\beta}^{[0]} = -\boldsymbol{\beta}$ and so $\sigma(\boldsymbol{\beta}^{[n]} \odot \boldsymbol{\rho}) = -\sigma(\boldsymbol{\beta}^{[0]} \odot \boldsymbol{\rho})$. Secondly, if n is odd, then the polarity word $\boldsymbol{\beta}$, which is a sequence of alternating '+1's and '−1's, starts and ends with the same symbol, i.e., $\beta_1 = \beta_n$. Therefore, the corresponding random walk (i, σ_i) starts and ends both with either an ascent ($\sigma_1 > \sigma_0$ and $\sigma_n > \sigma_{n-1}$) or a descent ($\sigma_1 < \sigma_0$ and $\sigma_n < \sigma_{n-1}$).

First consider the case $\sigma_0 \geq 0$ and split it into three sub-cases: $0 \leq \sigma_0 < d'$, $d' \leq \sigma_0 \leq k'$ and $\sigma_0 > k'$. For each of these cases, σ_1 ($\sigma_0 \to \sigma_1$) can be an ascent or a descent.

- $0 \leq \sigma_0 < d'$: This case is depicted graphically in Fig. 6.

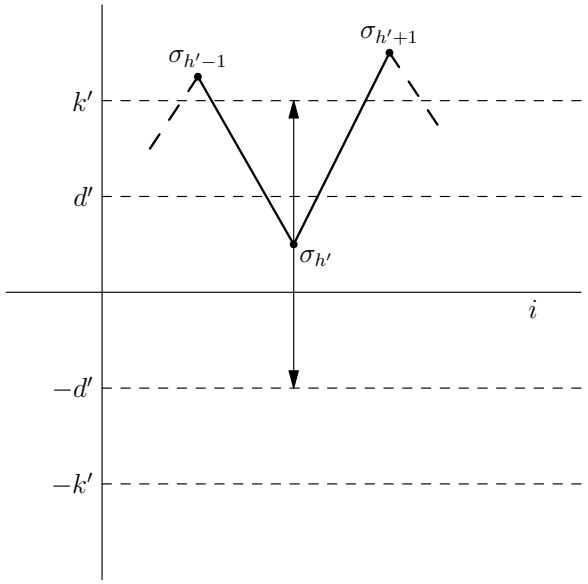

Fig. 3. First scenario when a random walk enters the inner region: $-d' < \sigma_{h'} \leq k'$ and $\sigma_{h'+1} > k'$.

1) Initial descent ($\sigma_1 < \sigma_0$): $\sigma_0 - 2k' \leq \sigma_1 \leq \sigma_0 - 2d'$. We split this into two further sub-cases:
   a) $-k' \leq \sigma_1 \leq \sigma_0 - 2d' < -d'$: σ_1 is a valley in the negative primary region, therefore $i = 1$ is a balancing index.
   b) $\sigma_1 < -k'$: the random walk has exited the inner region. σ_n is a descent since σ_1 is a descent. Consider the following two cases for the final descent:
      i) $d' < \sigma_n + 2d' \leq \sigma_{n-1} \leq k'$: σ_{n-1} is a peak in the positive primary region, therefore

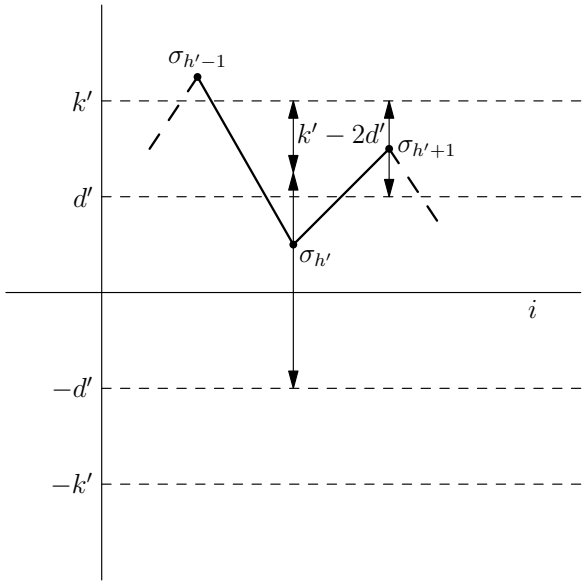

Fig. 4. Second scenario when a random walk enters the inner region: $-d' < \sigma_{h'} \leq k' - 2d'$ and $d' \leq \sigma_{h'+1} \leq k'$.

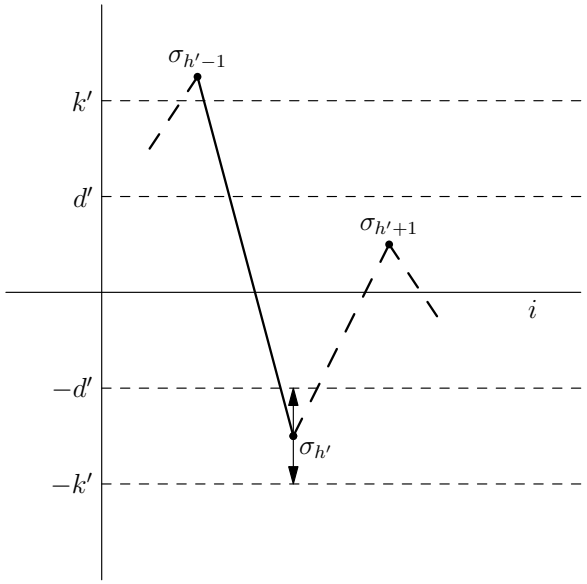

Fig. 5. Third scenario when a random walk enters the inner region: $-k' \leq \sigma_{h'} \leq -d'$.

$i = n-1$ is a balancing index.

ii) $\sigma_{n-1} > k'$: since $\sigma_1 < -k'$, then by Lemma 1, there exists at least one balancing index i, $1 < i < n-1$.

2) Initial ascent ($\sigma_1 > \sigma_0$): $\sigma_0 + 2d' \leq \sigma_1 \leq \sigma_0 + 2k'$. We split this into two further sub-cases:

a) $d' < \sigma_0 + 2d' \leq \sigma_1 \leq k'$: σ_1 is a peak in the positive primary region, therefore $i = 1$ is a balancing index.

b) $\sigma_1 > k'$: the random walk has exited the inner region. σ_n is an ascent since σ_1 is an ascent. Consider the following two cases for the final ascent:

i) $-k' \leq \sigma_{n-1} \leq \sigma_n - 2d' < -d'$: σ_{n-1} is a valley in the negative primary region, therefore $i = n-1$ is a balancing index.

ii) $\sigma_{n-1} < -k'$: since $\sigma_1 > k'$, then by Lemma 1, there exists at least one balancing index i, $1 < i < n-1$.

- $d' \leq \sigma_0 \leq k'$: This case is depicted graphically in Fig. 7.
  1) Initial descent ($\sigma_1 < \sigma_0$): σ_0 is a peak in the positive primary region, so $i = 0$ is a balancing index.
  2) Initial ascent ($\sigma_1 > \sigma_0$): $\sigma_0 + 2d' \leq \sigma_1 \leq \sigma_0 + 2k'$. We split this into two further sub-cases:
     a) $d' < \sigma_0 + 2d' \leq \sigma_1 \leq k'$: σ_1 is a peak in the positive primary region, therefore $i = 1$ is a balancing index.
     b) $\sigma_1 > k'$: the random walk has exited the inner region. σ_n is an ascent since σ_1 is an ascent. Consider the following two cases for the final ascent:
        i) $-k' \leq \sigma_{n-1} \leq \sigma_n - 2d' < \sigma_n \leq -d'$: σ_{n-1} is a valley in the negative primary region, therefore $i = n-1$ is a balancing index.
        ii) $\sigma_{n-1} < -k'$: since $\sigma_1 > k'$, then by Lemma 1, there exists at least one balancing index i, $1 < i < n-1$.
- $\sigma_0 > k'$: Since $\sigma_n = -\sigma_0 < -k'$, by Lemma 1, there exists at least one balancing index i, $0 < i < n$.

By symmetry the cases $-d' < \sigma_0 \leq 0$, $-k' \leq \sigma_0 \leq -d'$ and $\sigma_0 < -k'$ are equivalent to $0 \leq \sigma_0 < d'$, $d' \leq \sigma_0 \leq k'$ and $\sigma_0 > k'$, respectively. Therefore, for any (d', k')-RLL word, there exists at least one balancing index i, $0 \leq i \leq n$, if n is odd. ■

Corollary 1: For a RLL random walk (i, σ_i) corresponding to any (d', k')-RLL word $\boldsymbol{z} \Leftrightarrow \boldsymbol{\rho}^{(\beta_1)}$, there exists at least one index i, $0 \leq i \leq n$, such that $-(k'-d') \leq \sigma(\boldsymbol{\beta'}^{[i]} \odot \boldsymbol{\rho'}) \leq k'-d'$, where $\rho' = d'$, if n is odd.

Since a balancing index corresponds to a random walk peak in the positive primary region or a valley in the negative primary region, for a balancing index i we have that $\sigma(\boldsymbol{\beta}^{[i]} \odot \boldsymbol{\rho}) \in \mathcal{L}_{(d',k')}$ or $\sigma(\boldsymbol{\beta}^{[i]} \odot \boldsymbol{\rho}) \in -\mathcal{L}_{(d',k')}$, respectively. If $\rho' = d'$, in the first case we have $\sigma(\boldsymbol{\beta'}^{[i]} \odot \boldsymbol{\rho'}) \in \mathcal{L}_{(d',k')} - d'$ and in the second $\sigma(\boldsymbol{\beta'}^{[i]} \odot \boldsymbol{\rho'}) \in -\mathcal{L}_{(d',k')} + d'$. Therefore, for a balancing index i, $-(k'-d') \leq \sigma(\boldsymbol{\beta'}^{[i]} \odot \boldsymbol{\rho'}) \leq k'-d'$.

Comment 1: When searching for a balancing index i, it suffices to search in the range $0 \leq i \leq n-1$ (or $1 \leq i \leq n$), if n is odd. This follows from the fact that if $i = 0$ is a balancing index, then $i = n$ is also a balancing index. Suppose that $i = 0$ is a balancing index, then $\sigma(\boldsymbol{\beta'}^{[0]} \odot \boldsymbol{\rho'}) = \sigma(\boldsymbol{\beta'} \odot \boldsymbol{\rho'}) = 0$, where $\rho' = |\sigma_0|$ and $\beta' = -\beta_1$. Hence, $\sigma(\boldsymbol{\beta'} \odot \boldsymbol{\rho'}) = \rho'\beta' + \sigma(\boldsymbol{\beta} \odot \boldsymbol{\rho}) = -|\sigma_0|\beta_1 + \sigma(\boldsymbol{\beta} \odot \boldsymbol{\rho}) = 0$. But $\sigma(\boldsymbol{\beta'}^{[n]} \odot \boldsymbol{\rho'}) = |\sigma_n|(-(-\beta_n)) + \sigma(\boldsymbol{\beta}^{[n]} \odot \boldsymbol{\rho}) = |\sigma_n|\beta_n + \sigma(\boldsymbol{\beta}^{[n]} \odot \boldsymbol{\rho}) = |\sigma_0|\beta_1 - \sigma(\boldsymbol{\beta} \odot \boldsymbol{\rho}) = -|\sigma_0|(-\beta_1) - \sigma(\boldsymbol{\beta} \odot \boldsymbol{\rho}) = -(-|\sigma_0|\beta_1 + \sigma(\boldsymbol{\beta} \odot \boldsymbol{\rho})) = 0$. Therefore $i = n$ is also a balancing index.

The results of Theorem 1 and Corollary 1, which require n, the number of runs, to be odd, are better suited to variable-length codes (for a fixed n, for $\boldsymbol{z} \Leftrightarrow \boldsymbol{\rho}^{(\beta_1)}$, the length of $\boldsymbol{z}$, m, varies depending on the runlengths in $\boldsymbol{\rho}$). Since fixed-length codes are desirable (for example, to limit error propagation),

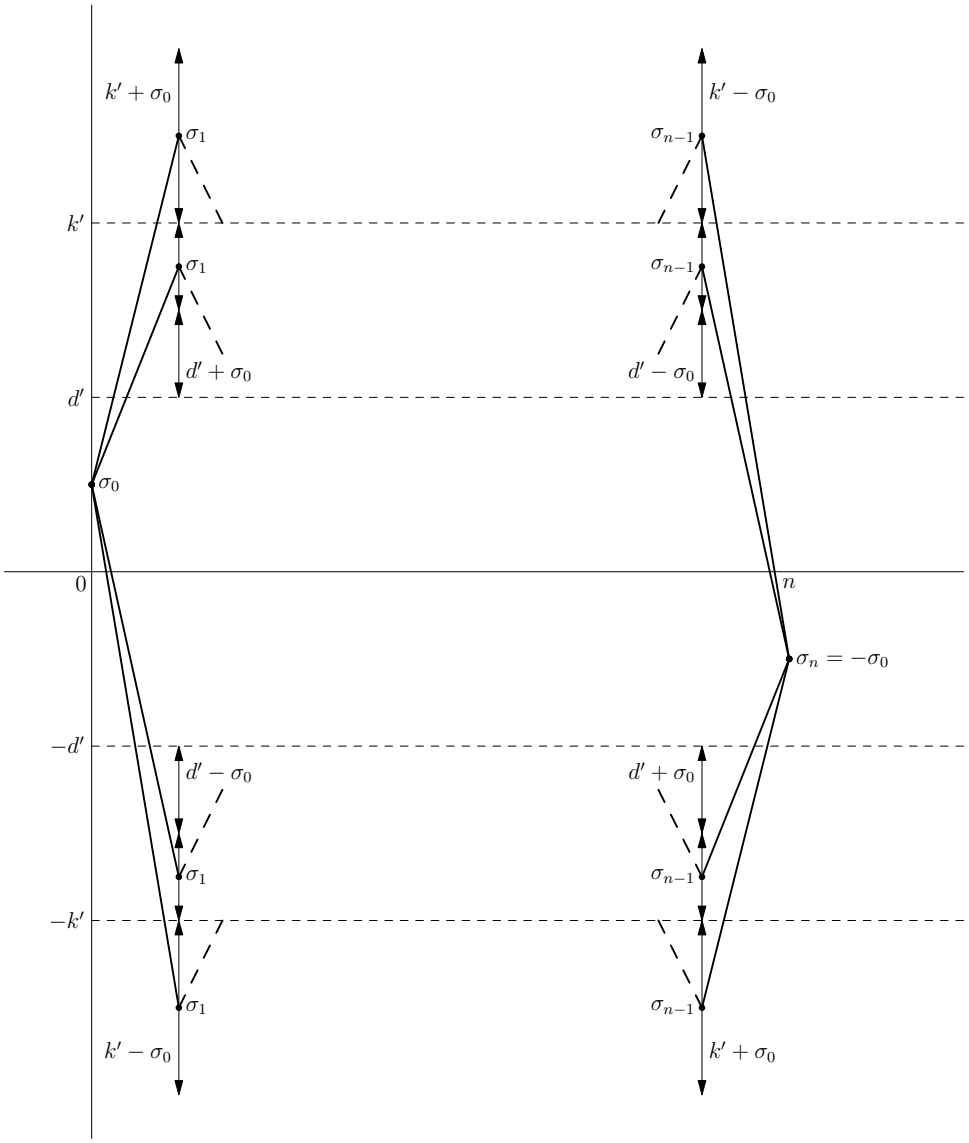

Fig. 6. A RLL random walk where $0 \leq \sigma_0 < d'$.

we are also interested in the case where m is fixed and n is variable. Therefore, we would like to remove the constraint that n is odd present in Theorem 1 and Corollary 1. This can in fact be achieved for Corollary 1 by substituting the constraint $k' \geq 2d'$ for the constraint that n is odd. This is proven in Theorem 2.

Theorem 2: For a RLL random walk (i, σ_i) corresponding to any (d', k')-RLL word $\boldsymbol{z} \Leftrightarrow \boldsymbol{\rho}^{(\beta_1)}$, there exists at least one index i, $0 \leq i \leq n$, such that $-(k'-d') \leq \sigma(\boldsymbol{\beta'}^{[i]} \odot \boldsymbol{\rho'}) \leq k'-d'$, where $\rho' = d'$, if $k' \geq 2d'$.

Proof: Any peak σ_i, $0 \leq \sigma_i \leq k'$, corresponds to an index i such that $-(k'-d') \leq \sigma(\boldsymbol{\beta'}^{[i]} \odot \boldsymbol{\rho'}) \leq k'-d'$, where $\rho' = d'$. In the case $d' \leq \sigma_i \leq k'$, if $\rho' = d'$, then $0 \leq \sigma_i - d' \leq k'-d'$ or $0 \leq \sigma(\boldsymbol{\beta'}^{[i]} \odot \boldsymbol{\rho'}) \leq k'-d'$. In the case $0 \leq \sigma_i < d'$, if $\rho' = d'$, then $-d' \leq \sigma_i - d' < 0$. However, since $k' \geq 2d'$, $k'-d' \geq d'$ and so $-(k'-d') \leq -d'$. Therefore $-(k'-d') \leq \sigma_i - d' < 0$ or $-(k'-d') \leq \sigma(\boldsymbol{\beta'}^{[i]} \odot \boldsymbol{\rho'}) < 0$. A similar argument shows that any valley σ_i, $-k' \leq \sigma_i \leq 0$, corresponds to an index i such that $-(k'-d') \leq \sigma(\boldsymbol{\beta'}^{[i]} \odot \boldsymbol{\rho'}) \leq k'-d'$, where $\rho' = d'$. Hence, the desired index i, $0 \leq i \leq n$, corresponds to a peak σ_i in the positive inner region (union of the positive primary and positive secondary regions) or a valley σ_i in the negative inner region (union of the negative primary and negative secondary regions).

The number of runs, n, can be even or odd. If n is odd, the result follows from Corollary 1. We only need to prove the result for even n. If n is even, an initial ascent ($\sigma_1 > \sigma_0$) in the RLL random walk is accompanied by a final descent ($\sigma_n < \sigma_{n-1}$), and vice versa. First consider the case $0 \leq \sigma_0 \leq k'$ (this scenario is depicted graphically in Fig. 8). If σ_1 is a descent, then σ_0 is a peak in the positive inner region, and thus for $i = 0$, $-(k'-d') \leq \sigma(\boldsymbol{\beta'}^{[0]} \odot \boldsymbol{\rho'}) \leq k'-d'$. Alternatively, if σ_1 is an ascent, then σ_n is a descent. Since $\sigma_n = -\sigma_0$, $-k' \leq \sigma_n \leq 0$, and so σ_n is a valley in the negative inner region, and thus for $i = n$, $-(k'-d') \leq \sigma(\boldsymbol{\beta'}^{[n]} \odot \boldsymbol{\rho'}) \leq k'-d'$. If $\sigma_0 > k'$, then $\sigma_n < -k'$, and so by Lemma 1, there exists

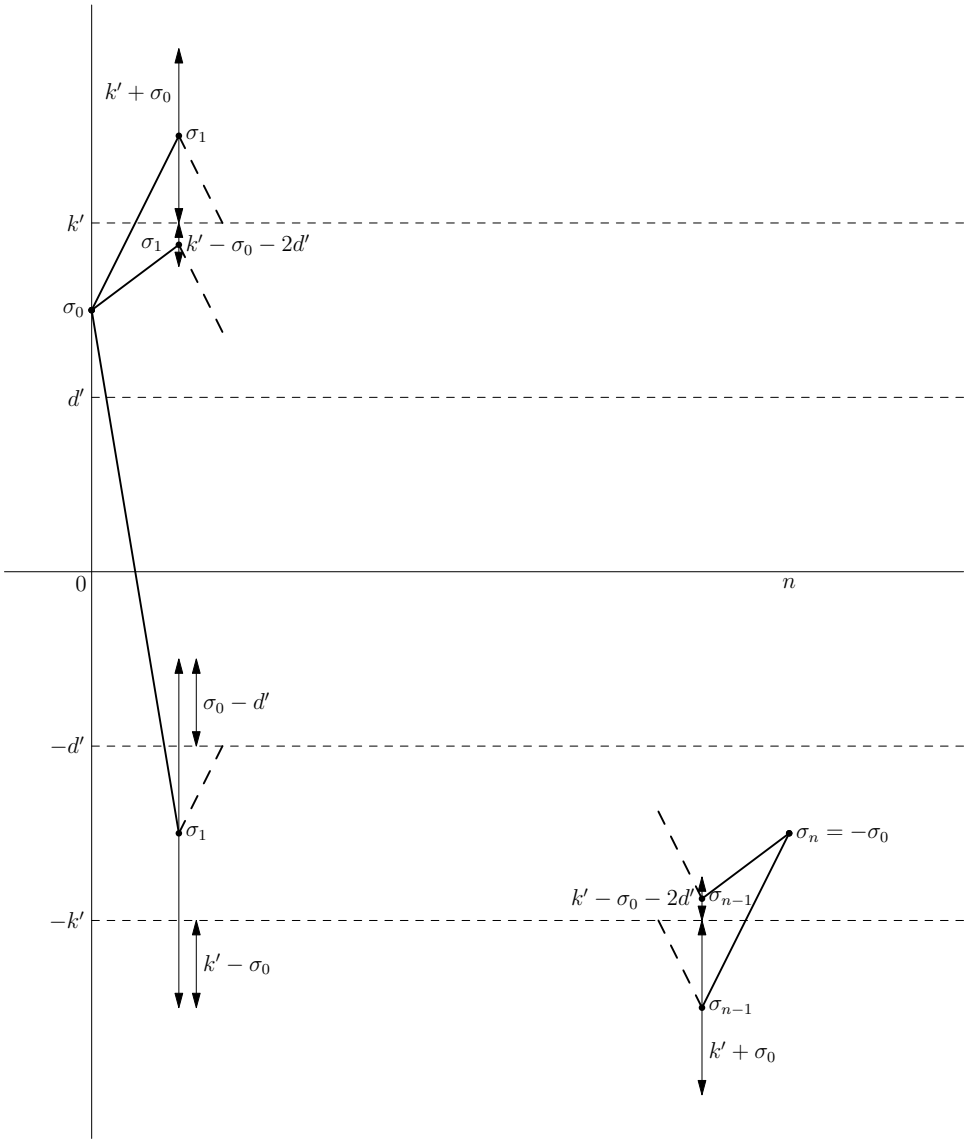

Fig. 7. A RLL random walk where $d' \leq \sigma_0 \leq k'$.

at least one balancing index i, $0 < i < n$. If $\rho' = d'$, then $-(k' - d') \leq \sigma(\boldsymbol{\beta'}^{[i]} \odot \boldsymbol{\rho'}) \leq k' - d'$.

By symmetry, the cases $-k' \leq \sigma_0 \leq 0$ and $\sigma_0 < -k'$ are equivalent to $0 \leq \sigma_0 \leq k'$ and $\sigma_0 > k'$, respectively. Therefore, for any (d', k')-RLL word, there exists at least one index i, $0 \leq i \leq n$, such that $-(k' - d') \leq \sigma(\boldsymbol{\beta'}^{[i]} \odot \boldsymbol{\rho'}) \leq k' - d'$, where $\rho' = d'$, if $k' \geq 2d'$. ■

For a (d', k')-RLL word, we will refer to any index i, $0 \leq i \leq n$, such that $-(k' - d') \leq \sigma(\boldsymbol{\beta'}^{[i]} \odot \boldsymbol{\rho'}) \leq k' - d'$ after the insertion of the interfix of length $\rho' = d'$, as a *near-balancing* index.

Theorem 1 and Corollary 1 form the basis of the variable-length balanced RLL codes presented in Section V. Theorem 2 forms the basis of the fixed-length balanced RLL codes also presented in Section V. Before presenting these codes and their associated encoding and decoding algorithms, we derive generating functions in the next section that are able to count the number of RLL words that have a particular unbalance. This is important in determining the minimum required suffix lengths and hence the total redundancy of the proposed codes.

IV. Generating Functions

In this section, we derive generating functions that count the exact number of (d', k')-RLL words of a specific length that have a particular unbalance. These derivations are based on a modification of the approach used in [17] to count the number of balanced RLL words. The presentation in this section is based on the symbolic methods of analytic combinatorics. For an excellent introduction to this topic, refer to Flajolet and Sedgewick [20].

For a *combinatorial class* $\mathcal{A}$ consisting of all objects of all sizes, let the corresponding generating function be denoted as

$$A(z) = \sum_{r=0}^{\infty} a_r z^r,$$

where a_r, the coefficient of z^r, is the number of objects in $\mathcal{A}$ of size r. The notation $[z^r]A(z)$ is used to extract the coefficient of z^r in $A(z)$, i.e., $[z^r]A(z) = a_r$.

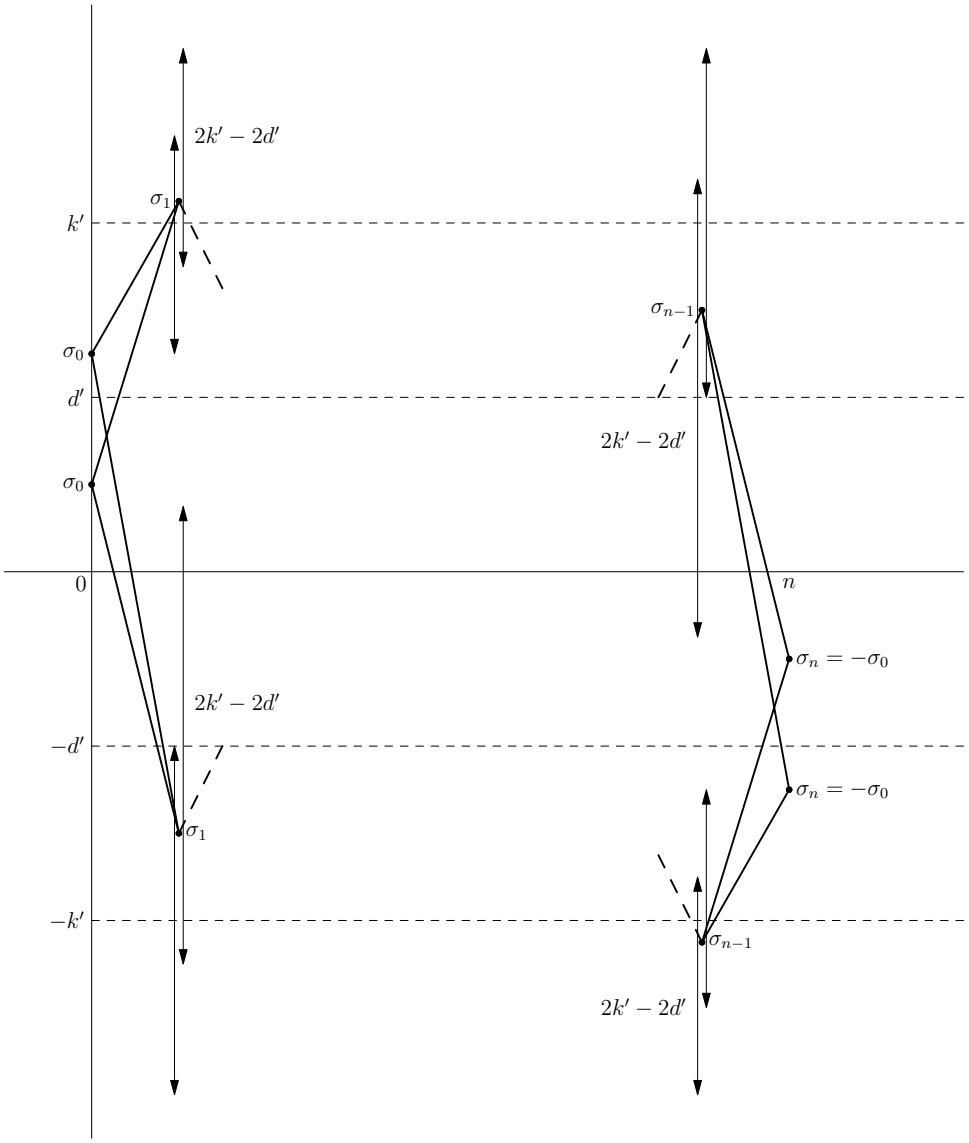

Fig. 8. A RLL random walk where $0 \leq \sigma_0 \leq k'$ and n is even.

Let the generating function

$$L_{(d',k')}(z) = z^{d'} + z^{d'+1} + \ldots + z^{k'}$$

correspond to a combinatorial class consisting of objects whose sizes are equal to the permissible lengths of runs in a (d', k')-RLL word. Then, for a combinatorial class consisting of objects constructed by any combination of objects whose size is at least d' and at most k', the generating function is

$$R_{(d',k')}(z) = \frac{1}{1 - L_{(d',k')}(z)} = \frac{1}{1 - z^{d'} - z^{d'+1} - \ldots - z^{k'}}.$$

We use bivariate generating functions to count the number of (d', k')-RLL words of length r that consist of n runs. A bivariate generating function $A(z, u)$ consists of two variables, z and u. As in the univariate case, z marks the object size. On the other hand, u is introduced to mark some auxiliary integer-valued parameter. The bivariate generating function $A(z, u)$ is defined as

$$A(z, u) = \sum_{r,n \geq 0} a_{r,n} z^r u^n.$$

In analogy with the univariate case, the notation $[z^r u^n]A(z, u)$ extracts the coefficient of $z^r u^n$ in $A(z, u)$, i.e., $[z^r u^n]A(z, u) = a_{r,n}$. We will use u to count the number of occurrences of the runlength-limited objects and so the bivariate generating function of interest is

$$\begin{aligned} R_{(d',k')}(z, u) &= \frac{1}{1 - uL_{(d',k')}(z)} \\ &= \frac{1}{1 - u(z^{d'} + z^{d'+1} + \ldots + z^{k'})}. \end{aligned}$$

The coefficient $[z^r u^n]R_{(d',k')}(z, u)$ gives the number of (d', k')-RLL words of length r that consist of n runs.

Given two generating functions $B(z) = \sum_{r \geq 0} b_r z^r$ and $C(z) = \sum_{r \geq 0} c_r z^r$, the Hadamard product $A(z) =$

$\sum_{r\geq 0} a_r z^r$ of $B(z)$ and $C(z)$, denoted as $A(z) = B(z) \odot C(z)$, has $a_r = b_r c_r$, i.e.,

$$A(z) = \sum_{r=0}^{\infty} b_r c_r z^r.$$

Similarly, for bivariate generating functions, the Hadamard product $A(z,u) = \sum_{r,n\geq 0} a_{r,n} z^r u^n$ of $B(z,u) = \sum_{r,n\geq 0} b_{r,n} z^r u^n$ and $C(z,u) = \sum_{r,n\geq 0} c_{r,n} z^r u^n$, denoted as $A(z,u) = B(z,u) \odot C(z,u)$, has $a_{r,n} = b_{r,n} c_{r,n}$, i.e.,

$$A(z,u) = \sum_{r,n\geq 0} b_{r,n} c_{r,n} z^r u^n.$$

Let $B^+(d',k',r)$ ($B^-(d',k',r)$) denote the number of balanced (d',k')-RLL words of length r, where r is even, that start with '+1' ('−1'). By inverting any balanced RLL word starting with '+1', a balanced RLL word starting with '−1' is obtained, and vice versa. Therefore, $B^+(d',k',r) = B^-(d',k',r)$. For convenience, we will denote this entity as

$$B(d',k',r) \triangleq B^+(d',k',r) = B^-(d',k',r).$$

A balanced RLL word can be constructed by interweaving two equal length RLL sub-words and assigning a positive polarity to one sub-word and a negative polarity to the other sub-word in the following two ways:

1) The two equal length sub-words consist of the same number of runs.
2) The two equal length sub-words differ by one in their number of runs.

Since $[z^r u^n] R_{(d',k')}(z,u)$ is the number of RLL words of length r consisting of n runs, and since any two RLL sub-words of equal length and equal number of runs can be interweaved together, for 1) there are $\{[z^r u^n] R_{(d',k')}(z,u)\}^2$ possibilities. This is true for all possible n given some r. This can be represented by

$$R_{(d',k')}(z,u) \odot R_{(d',k')}(z,u). \tag{4}$$

For 2), since any RLL sub-word of n runs and length r can be interweaved with any RLL sub-word of $n-1$ runs and length r, there are $\{[z^r u^n] R_{(d',k')}(z,u)\} \cdot \{[z^r u^{n-1}] R_{(d',k')}(z,u)\}$ possibilities. This is true for all possible n given some r. This can be represented by

$$R_{(d',k')}(z,u) \odot uR_{(d',k')}(z,u). \tag{5}$$

Summing (4) and (5), we have

$$\begin{aligned} B_{(d',k')}(z,u) = {} & R_{(d',k')}(z,u) \odot R_{(d',k')}(z,u) \\ & + R_{(d',k')}(z,u) \odot uR_{(d',k')}(z,u), \end{aligned} \tag{6}$$

and by replacing z with z^2 (which represents the combining of the two RLL sub-words into a balanced word which is twice the length of the sub-words) and setting $u = 1$ (thereby summing over the various possible number of runs for a given sub-word length), it follows that

$$B(d',k',2r) = [z^{2r}] B_{(d',k')}(z),$$

where $B_{(d',k')}(z) = B_{(d',k')}(z^2, 1)$. For further details, refer to Theorem 1 in [17].

Let $U^+(d',k',a,r)$ ($U^-(d',k',a,r)$) denote the number of (d',k')-RLL words of length r that start with '+1' ('−1') and have an unbalance of a, i.e., for a (d',k')-RLL word $\boldsymbol{z}$, $\sigma(\boldsymbol{z}) = a$. By inverting any RLL word of unbalance a starting with '+1', a RLL word of unbalance $-a$ starting with '−1' is obtained, and vice versa. Therefore, $U^+(d',k',a,r) = U^-(d',k',-a,r)$. Furthermore, let

$$U(d',k',a,r) \triangleq U^+(d',k',a,r) = U^-(d',k',-a,r).$$

Observe that a is odd (even) if, and only if, r is odd (even).

A RLL word of unbalance a can be constructed by interweaving two RLL sub-words differing in length by $|a|$ and consisting of the same number of runs or the number of runs differing by one. If $a > 0$, then the longer RLL sub-word is assigned a positive polarity and the shorter RLL sub-word a negative polarity, and vice versa if $a < 0$. Then,

$$U^+(d',k',a,r) = [z^r] U^+_{(d',k',a)}(z),$$

where

$$U^+_{(d',k',a)}(z) = U^+_{(d',k',a)}(z^2, 1) z^{-|a|} \tag{7}$$

and

$$\begin{aligned} U^+_{(d',k',a)}(z,u) = {} & R_{(d',k')}(z,u) \odot R_{(d',k')}(z,u) z^{|a|} \\ & + R_{(d',k')}(z,u) \odot uR_{(d',k')}(z,u) z^{|a|}, \end{aligned} \tag{8}$$

for $a > 0$. In (8), the second terms of the first and second Hadamard product, $R_{(d',k')}(z,u)$ and $uR_{(d',k')}(z,u)$, respectively, are multiplied by $z^{|a|}$ since the difference in lengths of the two RLL sub-words is $|a|$. Multiplication by $z^{-|a|}$ in (7) is to compensate for multiplication by $z^{|a|}$ in (8).

Similarly, for $a < 0$,

$$\begin{aligned} U^+_{(d',k',a)}(z,u) = {} & R_{(d',k')}(z,u) z^{|a|} \odot R_{(d',k')}(z,u) \\ & + R_{(d',k')}(z,u) z^{|a|} \odot uR_{(d',k')}(z,u). \end{aligned} \tag{9}$$

Notice that both (8) and (9) reduce to (6) when $a = 0$.

In Construction 3 from [7], to ensure that the runlength-constraints are maintained at the boundary between the suffix and the RLL word modified by Knuth's balancing method plus interfix, the first run of the suffix has to be of length at least d' and at most $k' - d' + 1$. Furthermore, the last run in the suffix need not satisfy the minimum runlength constraint.

Let $\hat{U}^+(d',k',a,r)$ ($\hat{U}^-(d',k',a,r)$) denote the number of (d',k')-RLL words of length r that start with '+1' ('−1'), have an unbalance of a, i.e., for a (d',k')-RLL word $\boldsymbol{z}$, $\sigma(\boldsymbol{z}) = a$, whose first runlength is at least d' and at most $k' - d' + 1$ and whose last run need not obey the minimum runlength constraint. By inverting any RLL word of unbalance a starting with '+1' and satisfying the prescribed conditions on the first and last run, a RLL word of unbalance $-a$ starting with '−1' and satisfying the same conditions on the first and last run is obtained, and vice versa. Therefore, $\hat{U}^+(d',k',a,r) = \hat{U}^-(d',k',-a,r)$. Furthermore, let

$$\hat{U}(d',k',a,r) \triangleq \hat{U}^+(d',k',a,r) = \hat{U}^-(d',k',-a,r).$$

Then,

$$\hat{U}^+(d',k',a,r) = [z^r] \hat{U}^+_{(d',k',a)}(z),$$

where

$$\hat{U}^{+}_{(d',k',a)}(z) = \hat{U}^{+}_{(d',k',a)}(z^2,1)z^{-|a|} \tag{10}$$

and, for $a > 0$,

$$\hat{U}^{+}_{(d',k',a)}(z,u) = \tilde{R}_{(d',k')}(z,u) \odot \bar{R}_{(d',k')}(z,u)z^{|a|} + \tilde{\bar{R}}_{(d',k')}(z,u) \odot uR_{(d',k')}(z,u)z^{|a|}, \tag{11}$$

where

$$\tilde{L}_{(d',k')}(z) = z^{d'} + z^{d'+1} + \ldots + z^{k'-d'+1}$$

and

$$\tilde{R}_{(d',k')(z,u)} = \frac{u\tilde{L}_{(d',k')}(z)}{1 - uL_{(d',k')}(z)}$$

and

$$\bar{L}_{(<d')}(z) = 1 + z + z^2 + \ldots + z^{d'-1}$$

and

$$\bar{R}_{(d',k')}(z,u) = \frac{1 + u(\bar{L}_{(<d')}(z) - 1)}{1 - uL_{(d',k')}(z)}$$

and

$$\tilde{\bar{R}}_{(d',k')}(z,u) = \frac{u\tilde{L}_{(d',k')}(z)\big[1 + u(\bar{L}_{(<d')}(z) - 1)\big]}{1 - uL_{(d',k')}(z)}.$$

Similarly, for $a < 0$,

$$\hat{U}^{+}_{(d',k',a)}(z,u) = \tilde{R}_{(d',k')}(z,u)z^{|a|} \odot \bar{R}_{(d',k')}(z,u) + \tilde{\bar{R}}_{(d',k')}(z,u)z^{|a|} \odot uR_{(d',k')}(z,u). \tag{12}$$

With these generating functions it is easy to determine the number of balancing or near-balancing indices that can be represented by a suffix of a particular length.

V. Encoding and Decoding Algorithms

In this section, we formalize the various encoding and decoding algorithms of variable- and fixed-length balanced RLL codes based on Theorem 1, Corollary 1 and Theorem 2. Here, we also present a runlength constraint violating marker, where the violating run is used to indicate the balancing index, based on Theorem 1 and RLL Knuth-like balancing procedure introduced in Section III. It is assumed for all the codes presented in this section, that a RLL word or sequence that acts as the input to the RLL Knuth-like balancing method is produced by a prior art method. In this section we adopt the notational convention of using a tilde over a length variable to signify word lengths which are variable. Absence of a tilde then signifies word lengths which are fixed.

A. Balanced RLL Codes with Stringent Runlength Constraints

All the codes presented in this sub-section adhere to stringent runlength constraints, i.e., no runlength constraint violations are permitted. After the production of a balanced or a near-balanced RLL word using the RLL Knuth-like balancing method and inserting an appropriate interfix, a suffix is concatenated with this balanced or near-balanced RLL word. This suffix performs two important functions:

1) Encodes the balancing index so that the decoder is able to extract the balancing index from the suffix and thereby remove the interfix and undo the inversion of the portion of the original RLL word caused by the RLL Knuth-like balancing method.
2) In the case of a near-balanced RLL word, the suffix compensates for the unbalance of the near-balanced word and thereby produces an overall word that is balanced.

Let $\mathcal{U}^{(p)}(d',k',a,r)$ represent the set of all bipolar (d',k')-RLL words $\boldsymbol{s} = (s_1, s_2, \ldots, s_r) \in \{-1,+1\}^r$ with $s_1 = p$ such that $\sigma(\boldsymbol{s}) = a$. Then $|\mathcal{U}^{(+)}(d',k',a,r)| = |\mathcal{U}^{(-)}(d',k',-a,r)| = U^{+}(d',k',a,r) = [z^r]U^{+}_{(d',k',a)}(z)$. Note that when $a = 0$, $U^{+}(d',k',0,r) = B^{+}(d',k',r) = B(d',k',r) = [z^r]B_{(d',k')}(z)$.

The bijective mapping

$$\boldsymbol{\psi}^{(p)} : \mathcal{L}_{(-k'+d',k'-d')} \times \{0,1,\ldots,\nu\} \to \mathcal{U}^{(p)}(d',k',a,r)$$

associates each possible balancing index with a unique RLL word from $\mathcal{U}^{(p)}(d',k',a,r)$ for each $-(k'-d') \le a \le k'-d'$, where $\boldsymbol{\psi}^{(+)}(a,i) = -\boldsymbol{\psi}^{(-)}(-a,i)$ and $\mathcal{L}_{(-k'+d',k'-d')} \triangleq (\mathcal{L}_{(d',k')} - d') \cup (-\mathcal{L}_{(d',k')} + d')$. Note that $\nu = n-1$ for the variable length codes and $\nu = n$ for the fixed length codes. Hence, the mapping $\boldsymbol{\psi}^{(p)}$ produces the desired suffix for a given unbalance a and (near-)balancing index i. The inverse mapping $[\boldsymbol{\psi}^{(p)}]^{-1}(\boldsymbol{s})$ produces the pair (a,i) from the suffix $\boldsymbol{s}$.

1) Variable-Length Codes 1 (VL1): In this variable-length code construction, the output of the RLL Knuth-like balancing method with a variable-length interfix is a balanced RLL word. Therefore, the suffix also needs to be a balanced RLL word. Since there are n possible balancing indices (see Comment 1), the length of the suffix is the minimum r such that

$$n \le B(d',k',r).$$

Suppose that $\boldsymbol{z} = (z_1, z_2, \ldots, z_{l-1}, z_l, \ldots)$, $z_l \in \{-1,+1\}$, denotes some (very) long bipolar (d',k')-RLL sequence. Partition $\boldsymbol{z}$ into variable-length sub-sequences $\boldsymbol{z}_j$, $j = 1,2,\ldots$, each consisting of n runs, where n is odd. Then

$$\boldsymbol{z} = (\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_{j-1}, \boldsymbol{z}_j, \ldots),$$

where

$$\boldsymbol{z}_j = (z_{j,1}, z_{j,2}, \ldots, z_{j,\tilde{m}}) \Leftrightarrow \boldsymbol{\rho}_j^{(\beta_{j,1})}$$

with $\boldsymbol{\rho}_j = (\rho_{j,1}, \rho_{j,2}, \ldots, \rho_{j,n})$, polarity word $\boldsymbol{\beta}_j = (\beta_{j,1}, \beta_{j,2}, \ldots, \beta_{j,n})$ and

$$\tilde{m} = \sum_{l=1}^{n} \rho_{j,l}.$$

For each variable-length word $\boldsymbol{z}_j$ of length $\tilde{m}$ preform the following encoding algorithm:

1) Apply the RLL Knuth-like balancing method from Section III to $\boldsymbol{z}_j \Leftrightarrow \boldsymbol{\rho}_j^{(\beta_{j,1})}$ and find a balancing index i, $0 \le i \le n-1$ (see Comment 1), i.e., an index i such that σ_i is a peak in the positive primary region or a valley in the negative primary region. Then, inserted a single-run interfix of length $\rho'_j = |\sigma(\boldsymbol{\beta}_j^{[i]} \odot \boldsymbol{\rho}_j)|$, $d' \le \rho'_j \le k'$,

into $\boldsymbol{\rho}_j$ to obtain $\boldsymbol{\rho}'_j$ as per (1), and insert β'_j as defined in (2) to obtain $\boldsymbol{\beta}'^{[i]}_j$ as per (3). Then $\boldsymbol{z}'_j \Leftrightarrow \boldsymbol{\rho}'^{(\beta'_{j,1})}_j$ is balanced, i.e., $\sigma(\boldsymbol{z}'_j) = \sigma(\boldsymbol{\beta}'^{[i]}_j \odot \boldsymbol{\rho}'_j) = 0$. The existence of at least one such balancing index is guaranteed by Theorem 1.

2) Generate the balanced RLL suffix corresponding to the balancing index i, i.e., $\boldsymbol{s}_j = (s_{j,1}, s_{j,2}, \ldots, s_{j,r}) = \boldsymbol{\psi}^{(-\beta_{j,n})}(0,i) \Leftrightarrow \boldsymbol{\tau}_j^{(\gamma_{j,1})}$, where $\boldsymbol{\tau}_j = (\tau_{j,1}, \tau_{j,2}, \ldots, \tau_{j,\tilde{t}})$, $\boldsymbol{\gamma}_j = (\gamma_{j,1}, \gamma_{j,2}, \ldots, \gamma_{j,\tilde{t}})$ is the polarity word and

$$r = \sum_{l=1}^{\tilde{t}} \tau_{j,l}.$$

Note that $\sigma(\boldsymbol{s}_j) = \sigma(\boldsymbol{\gamma}_j \odot \boldsymbol{\tau}_j) = 0$. Also, note that $\gamma_{j,1} = -\beta'_{j,n+1} = -\beta_{j,n}$, where $\beta'_{j,n+1}$ is the last symbol in $\boldsymbol{\beta}'^{[i]}_j$, in order to insure that the boundary between $\boldsymbol{z}'_j$ and $\boldsymbol{s}_j$ is also a run boundary.

3) Create the balanced RLL word $\boldsymbol{b}_j = (\boldsymbol{z}'_j, \boldsymbol{s}_j) \Leftrightarrow (\boldsymbol{\rho}'^{(\beta'_{j,1})}_j, \boldsymbol{\tau}_j^{(\gamma_{j,1})})$ by appending the suffix $\boldsymbol{s}_j$ to the balanced RLL word $\boldsymbol{z}'_j \Leftrightarrow \boldsymbol{\rho}'^{(\beta'_{j,1})}_j$. Note that $\sigma(\boldsymbol{b}_j) = \sigma(\boldsymbol{z}'_j) + \sigma(\boldsymbol{s}_j) = \sigma(\boldsymbol{\beta}'^{[i]}_j \odot \boldsymbol{\rho}'_j) + \sigma(\boldsymbol{\gamma}_j \odot \boldsymbol{\tau}_j) = 0$. If $\beta'_{j,1} = -s_{j-1,r}$, send $\boldsymbol{c}_j = \boldsymbol{b}_j$, else if $\beta'_{j,1} = s_{j-1,r}$, then send the inversion of $\boldsymbol{b}_j$, i.e., $\boldsymbol{c}_j = -\boldsymbol{b}_j \Leftrightarrow (\boldsymbol{\rho}'^{(-\beta'_{j,1})}_j, \boldsymbol{\tau}_j^{(-\gamma_{j,1})})$. This is done to ensure that the boundary between $\boldsymbol{c}_{j-1}$ and $\boldsymbol{c}_j$ is also a run boundary, allowing $\boldsymbol{c}_{j-1}$ and $\boldsymbol{c}_j$ to be cascaded (concatenated) without violating the runlength constraints.

Therefore, the output of the encoder corresponding to $\boldsymbol{z}$ is

$$\begin{aligned}\boldsymbol{c} &= (\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_{j-1}, \boldsymbol{c}_j, \ldots) \\ &= (c_1, c_2, \ldots, c_{l-1}, c_l, \ldots).\end{aligned}$$

This is also the input to the decoder. Although $\boldsymbol{c}_j$, $j = 1, 2, \ldots$, are variable length, the decoder can deduce the codeword boundaries by partitioning the sequence $\boldsymbol{c} = (c_1, c_2, \ldots, c_{l-1}, c_l, \ldots)$ into sub-sequences $\boldsymbol{c}_j = (\boldsymbol{z}'_j, \boldsymbol{s}_j)$, $j = 1, 2, \ldots$, where the first portion of $\boldsymbol{c}_j$, $\boldsymbol{z}'_j$, consists of $n+1$ runs and the last portion of $\boldsymbol{c}_j$, $\boldsymbol{s}_j$, is of length r.

For each variable-length balanced RLL codeword $\boldsymbol{c}_j$ perform the following decoding algorithm:

1) Decode the balancing index i, $0 \le i \le n-1$, from the fixed-length suffix $\boldsymbol{s}_j$ as $(0,i) = [\boldsymbol{\psi}^{(s_{j,1})}]^{-1}(\boldsymbol{s}_j)$.

2) For $\boldsymbol{z}'_j \Leftrightarrow \boldsymbol{\rho}'^{(\beta'_{j,1})}_j$, with $\boldsymbol{\rho}'_j = (\rho'_{j,1}, \rho'_{j,2}, \ldots, \rho'_{j,n+1})$ and a polarity word $\boldsymbol{\beta}'_j = (\beta'_{j,1}, \beta'_{j,2}, \ldots, \beta'_{j,n+1})$, obtain $\boldsymbol{\rho}_j$ from $\boldsymbol{\rho}'_j$ by deleting the interfix $\rho'_{j,i+1}$ at index $i+1$, i.e.,

$$\boldsymbol{\rho}_j = (\rho'_{j,1}, \rho'_{j,2}, \ldots, \rho'_{j,i-1}, \rho'_{j,i}, \rho'_{j,i+2}, \ldots, \rho'_{j,n+1}).$$

Also obtain $\boldsymbol{\beta}_j$ from $\boldsymbol{\beta}'_j$ by deleting $\beta'_{j,i+1}$ at index $i+1$ and inverting all $\beta'_{j,l}$ for $l = 1, 2, \ldots, i$ (thereby undoing the inversion introduced during the RLL balancing method), i.e.,

$$\boldsymbol{\beta}_j = (-\beta'_{j,1}, -\beta'_{j,2}, \ldots, -\beta'_{j,i-1}, -\beta'_{j,i}, \beta'_{j,i+2}, \ldots, \beta'_{j,n+1}).$$

Then the original RLL word is $\boldsymbol{z}_j = (z_{j,1}, z_{j,2}, \ldots, z_{j,\tilde{m}}) \Leftrightarrow \boldsymbol{\rho}_j^{(\beta_{j,1})}$, if $\beta_{j,1} = -z_{j-1,\tilde{m}}$, and $\boldsymbol{z}_j \Leftrightarrow \boldsymbol{\rho}_j^{(-\beta_{j,1})}$, if $\beta_{j,1} = z_{j-1,\tilde{m}}$, where $\beta_{j,1}$ is the first symbol in $\boldsymbol{\beta}_j$.

If the source RLL word is already balanced, i.e., $\sigma(\boldsymbol{z}_j) = 0$, the above encoding and decoding algorithms can be modified to achieve slightly better average code rate. In such a case, an interfix is not needed. If a sequence consisting of $n+1$ runs is balanced, it is clear that the sub-sequence consisting of the first n runs cannot be balanced. Similarly, if a sequence of n runs is balanced, then the addition of any run to this sequence results in an unbalanced sequence. Therefore, the decoder can uniquely distinguish between whether the original source RLL word was balanced or not. The decoder first extracts the first n runs. If this is a balanced word, no interfix was added and this is the original RLL word. Otherwise, an interfix was added, and the same decoding procedure as described above is executed. It should be obvious that if the source RLL word is already balanced, the appending of a suffix is unnecessary, resulting in a further average code rate improvement.

Example 2: Consider the case $d' = 2$, $k' = 4$ and $n = 5$. Then

$$B_{(2,4)}(z) = 1 + z^4 + z^6 + 3z^8 + 4z^{10} + 13z^{12} + \ldots$$

Therefore, since $n \le B(2,4,r)$, the minimum suffix length is $r = 12$. Table I gives a possible manifestation of the mapping $\boldsymbol{\psi}^{(+)}$ (recall that $\boldsymbol{\psi}^{(-)}(0,i) = -\boldsymbol{\psi}^{(+)}(0,i)$).

TABLE I
THE MAPPING $\boldsymbol{\psi}^{(+)}(0,i)$ FOR EXAMPLE 2.

i	$\boldsymbol{\psi}^{(+)}(0,i)$
0	$(++--++++---) \Leftrightarrow (2,2,4,4)^{(+)}$
1	$(++---++++--) \Leftrightarrow (2,3,4,3)^{(+)}$
2	$(++----++++--) \Leftrightarrow (2,4,4,2)^{(+)}$
3	$(+++--++++---) \Leftrightarrow (3,2,3,4)^{(+)}$
4	$(+++--++++--) \Leftrightarrow (3,3,3,3)^{(+)}$

Suppose that some $(2,4)$-RLL encoder produces the $(2,4)$-RLL sequence

$$\boldsymbol{z} = (----++++--++++----++++---++++\ldots),$$

where

$$\boldsymbol{z} \Leftrightarrow \boldsymbol{\rho}^{(\beta_1)} = (4,3,2,4,3,2,3,4,2,4,\ldots)^{(-)}.$$

Partition $\boldsymbol{z}$ into sub-sequences consisting of $n = 5$ runs. Then $\boldsymbol{z} = (\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots)$ where $\boldsymbol{z}_1 \Leftrightarrow \boldsymbol{\rho}_1^{(\beta_{1,1})} = (4,3,2,4,3)^{(-)}$ and $\boldsymbol{z}_2 \Leftrightarrow \boldsymbol{\rho}_2^{(\beta_{2,1})} = (2,3,4,2,4)^{(+)}$.

Encode $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ as follows. For $\boldsymbol{z}_1$, $i = 0, 3, 4, 5$ are balancing indices (cf. Example 1). We will use the first balancing index $i = 0$. The length of the interfix run is $\rho' = |\sigma_0| = 2$, and so $\boldsymbol{\rho}'_1 = (2,4,3,2,4,3)$ and $\boldsymbol{\beta}'^{[0]}_1 = (+-+-+-)$. It is easy to verify that, with $\boldsymbol{z}'_1 \Leftrightarrow \boldsymbol{\rho}'^{(+)}_1$, $\sigma(\boldsymbol{z}'_1) = \sigma(\boldsymbol{\beta}'^{[0]}_1 \odot \boldsymbol{\rho}'_1) = 0$. Then, $\boldsymbol{s}_1 = \boldsymbol{\psi}^{(+)}(0,0) = (++--++++---) \Leftrightarrow (2,2,4,4)^{(+)} = \boldsymbol{\tau}_1^{(+)}$, and so $\boldsymbol{c}_1 = (\boldsymbol{z}'_1, \boldsymbol{s}_1)$.

For $\boldsymbol{z}_2$, as can be easily verified, $i = 4$ is the only balancing index. The length of the interfix run is $\rho' = |\sigma_4| = 3$, and so $\boldsymbol{\rho}'_2 = (2, 3, 4, 2, 3, 4)$ and $\boldsymbol{\beta}'^{[4]}_2 = (- + - + - +)$. It is easy to verify that, with $\boldsymbol{z}'_2 \Leftrightarrow \boldsymbol{\rho}'^{(-)}_2$, $\sigma(\boldsymbol{z}'_2) = \sigma(\boldsymbol{\beta}'^{[4]}_2 \odot \boldsymbol{\rho}'_2) = 0$. Then, $\boldsymbol{s}_2 = \boldsymbol{\psi}^{(-)}(0, 4) = -\boldsymbol{\psi}^{(+)}(0, 4) = (- - - + + + - - - + + +) \Leftrightarrow (3, 3, 3, 3)^{(-)} = \boldsymbol{\tau}^{(-)}_2$. However, since $s_{1,12} = -1 = \beta'_{2,1}$, $\boldsymbol{c}_2 = (-\boldsymbol{z}'_2, -\boldsymbol{s}_2) \Leftrightarrow (\boldsymbol{\rho}'^{(+)}_2, \boldsymbol{\tau}^{(+)}_2)$.

Therefore, the output of the VL1 encoder is

$$\boldsymbol{c} = (\boldsymbol{c}_1, \boldsymbol{c}_2) \Leftrightarrow (2, 4, 3, 2, 4, 3, 2, 2, 4, 4, 2, 3, 4, 2, 3, 4, 3, 3, 3, 3, \ldots)^{(+)}.$$

At the VL1 decoder, the first 6 runs are $\boldsymbol{z}'_1$, the next 12 symbols are $\boldsymbol{s}_1$, the next 6 runs are $\boldsymbol{z}'_2$ and the next 12 symbols are $\boldsymbol{s}_2$. Thus

$$\boldsymbol{z}'_1 \Leftrightarrow \boldsymbol{\rho}'^{(+)}_1 = (2, 4, 3, 2, 4, 3)^{(+)}$$
$$\boldsymbol{s}_1 \Leftrightarrow (2, 2, 4, 4)^{(+)}$$

and

$$\boldsymbol{z}'_2 \Leftrightarrow \boldsymbol{\rho}'^{(+)}_2 = (2, 3, 4, 2, 3, 4)^{(+)}$$
$$\boldsymbol{s}_2 \Leftrightarrow (3, 3, 3, 3)^{(+)}.$$

For $\boldsymbol{c}_1 = (\boldsymbol{z}'_1, \boldsymbol{s}_1)$, the balancing index $i = 0$ is obtained from $(0, 0) = [\boldsymbol{\psi}^{(+)}]^{-1}(\boldsymbol{s}_1)$. Then delete the interfix run at index $i + 1 = 1$ from $\boldsymbol{\rho}'_1$ to obtain

$$\boldsymbol{z}_1 \Leftrightarrow \boldsymbol{\rho}^{(-)}_1 = (4, 3, 2, 4, 3)^{(-)}.$$

For $\boldsymbol{c}_2 = (\boldsymbol{z}'_2, \boldsymbol{s}_2)$, the balancing index $i = 4$ is obtained from $(0, 4) = [\boldsymbol{\psi}^{(+)}]^{-1}(\boldsymbol{s}_2)$. Then delete the interfix run at index $i + 4 = 5$ from $\boldsymbol{\rho}'_2$ and invert the polarity of the first $i = 4$ runs to obtain $\boldsymbol{\rho}^{(-)}_2 = (2, 3, 4, 2, 4)^{(-)}$. Since $\beta_{2,1} = z_{1,16} = -1$

$$\boldsymbol{z}_2 \Leftrightarrow \boldsymbol{\rho}^{(+)}_2 = (2, 3, 4, 2, 4)^{(+)}.$$

Therefore

$$\boldsymbol{z} = (\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots) \Leftrightarrow (4, 3, 2, 4, 3, 2, 3, 4, 2, 4, \ldots)^{(-)},$$

which corresponds to the original RLL sequence. □

2) Variable-Length Codes 2 (VL2): These variable-length balanced RLL codes are almost identical to VL1, the only difference being that while the interfix length ρ' in VL1 is variable, $d' \le \rho' \le k'$, in VL2 the interfix is fixed-length, $\rho' = d'$. By Corollary 1, even the near-balancing index i, $0 \le i \le n - 1$, is identical to the balancing index from VL1. With an interfix $\rho' = d'$, for $\boldsymbol{z}_j$ we have that $-(k' - d') \le \sigma(\boldsymbol{\beta}'^{[i]}_j \odot \boldsymbol{\rho}'_j) \le k' - d'$. Then, the suffix $\boldsymbol{s}_j$ needs to have an unbalance $\sigma(\boldsymbol{s}_j) = a = -\sigma(\boldsymbol{\beta}'^{[i]}_j \odot \boldsymbol{\rho}'_j)$. Note that the unbalance $\sigma(\boldsymbol{s}_j) = a$ is odd (even) if, and only if, the length r of the suffix $\boldsymbol{s}_j$ is odd (even). Also, since the length $\tilde{m}$ of $\boldsymbol{z}_j$ depends on $\boldsymbol{\rho}_j$, it may be even or odd. Thus, the length $\tilde{m} + d'$ of $\boldsymbol{z}'_j$ may be even or odd. In order for $(\boldsymbol{z}'_j, \boldsymbol{s}_j)$ to be balanced, $\tilde{m} + d' + r$ must be even, indicating that r is even (odd) whenever $\tilde{m} + d'$ is even (odd). If the suffix length r is even, then suffixes of odd length $r - 1$ (or $r + 1$) are also required. Then, since there are n possible near-balancing indices, the required minimum suffix length r, r even, must satisfy

$$n \le \min\Big\{U(d', k', a, r) : a \in \mathcal{L}_{(-k'+d', k'-d')}, a \text{ is even}\Big\}$$

and

$$n \le \min\Big\{U(d', k', a, r-1) : a \in \mathcal{L}_{(-k'+d', k'-d')}, a \text{ is odd}\Big\}$$

or

$$n \le \min\Big\{U(d', k', a, r+1) : a \in \mathcal{L}_{(-k'+d', k'-d')}, a \text{ is odd}\Big\}.$$

The encoding and decoding algorithms of VL2 are essentially identical to that of VL1 with minor modifications. In encoding step 1) from VL1, for VL2 the interfix is $\rho'_j = d'$ and $-(k' - d') \le \sigma(\boldsymbol{z}'_j) = \sigma(\boldsymbol{\beta}'^{[i]}_j \odot \boldsymbol{\rho}'_j) \le k' - d'$. In encoding step 2) from VL1, for VL2 the suffix is $\boldsymbol{s}_j = (s_{j,1}, s_{j,2}, \ldots, s_{j,\varsigma}) = \boldsymbol{\psi}^{(-\beta_{j,n})}(a, i) \Leftrightarrow \boldsymbol{\tau}^{(\gamma_{j,1})}_j$, where $a = -\sigma(\boldsymbol{\beta}'^{[i]}_j \odot \boldsymbol{\rho}'_j)$ and $\varsigma = r$ (even) if the length of $\boldsymbol{z}'_j$ is even or $\varsigma = r - 1$ (or $\varsigma = r + 1$) (odd) if the length of $\boldsymbol{z}'_j$ is odd. Note that $\sigma(\boldsymbol{b}_j) = \sigma(\boldsymbol{z}'_j) + \sigma(\boldsymbol{s}_j) = \sigma(\boldsymbol{\beta}'^{[i]}_j \odot \boldsymbol{\rho}'_j) + \sigma(\boldsymbol{\gamma}_j \odot \boldsymbol{\tau}_j) = -a + a = 0$. On the decoding side, the decoder can deduce the codeword boundaries by partitioning the sequence $\boldsymbol{c} = (c_1, c_2, \ldots, c_{l-1}, c_l, \ldots)$ into sub-sequences $\boldsymbol{c}_j = (\boldsymbol{z}'_j, \boldsymbol{s}_j)$, $j = 1, 2, \ldots$, where the first portion of $\boldsymbol{c}_j$, $\boldsymbol{z}'_j$, consists of $n+1$ runs and the last portion of $\boldsymbol{c}_j$, $\boldsymbol{s}_j$, is of length r if the length of $\boldsymbol{z}'_j$ is even, or of length $r - 1$ ($r + 1$) if the length of $\boldsymbol{z}'_j$ is odd. In decoding step 1) from VL1, for VL2 the extraction of the near-balancing index from $\boldsymbol{s}_j$ is $(a, i) = [\boldsymbol{\psi}^{(s_{j,1})}]^{-1}(\boldsymbol{s}_j)$.

3) Fixed-Length Codes (FL): For the fixed-length balanced RLL codes, $\boldsymbol{z}_j$ is of fixed length m, while the corresponding $\boldsymbol{\rho}_j$ is of variable length $\tilde{n}$. Hence, the number of runs $\tilde{n}$ in $\boldsymbol{z}_j$ may be even or odd. The code FL is based on Theorem 2, which guarantees the existence of at least one near-balancing index i, $0 \le i \le \tilde{n}$, such that $-(k' - d') \le \sigma(\boldsymbol{\beta}'^{[i]}_j \odot \boldsymbol{\rho}'_j) \le k' - d'$ if the interfix is of length $\rho'_j = d'$ and $k' \ge 2d'$. Then, the suffix $\boldsymbol{s}_j$ needs to have an unbalance $\sigma(\boldsymbol{s}_j) = a = -\sigma(\boldsymbol{\beta}'^{[i]}_j \odot \boldsymbol{\rho}'_j)$. For a (d', k')-RLL word of length m, the maximum number of runs is $\max\{\tilde{n}\} = \lfloor m/d' \rfloor$, where $\lfloor x \rfloor$ denotes the largest integer less than or equal to x. Since $m + d' + r$ must be even, if $m + d'$ is even (odd), then r, the length of the suffix, is even (odd). If r is even, then the length of the suffix is the minimum r such that

$$\left\lfloor \frac{m}{d'} \right\rfloor + 1 \le \min\Big\{U(d', k', a, r) : a \in \mathcal{L}_{(-k'+d', k'-d')}, a \text{ is even}\Big\},$$

and if r is odd, then the length of the suffix is the minimum r such that

$$\left\lfloor \frac{m}{d'} \right\rfloor + 1 \le \min\Big\{U(d', k', a, r) : a \in \mathcal{L}_{(-k'+d', k'-d')}, a \text{ is odd}\Big\}.$$

Suppose that $\boldsymbol{z} = (\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_{j-1}, \boldsymbol{z}_j, \ldots)$ is produced by a prior art (d', k')-RLL block code where the codeword boundaries correspond to run boundaries. Then

$$\boldsymbol{z}_j = (z_{j,1}, z_{j,2}, \ldots, z_{j,m}) \Leftrightarrow \boldsymbol{\rho}^{(\beta_{j,1})}_j$$

TABLE II
THE MAPPING $\boldsymbol{\psi}^{(+)}(1,i)$ FOR EXAMPLE 3.

i	$\boldsymbol{\psi}^{(+)}(1,i)$
0	$(++--++++---) \Leftrightarrow (2,2,4,3)^{(+)}$
1	$(++---++++--) \Leftrightarrow (2,3,4,2)^{(+)}$
2	$(+++--++++---) \Leftrightarrow (3,2,3,3)^{(+)}$
3	$(+++---++++--) \Leftrightarrow (3,3,3,2)^{(+)}$
4	$(++++--++---) \Leftrightarrow (4,2,2,3)^{(+)}$
5	$(++++---++--) \Leftrightarrow (4,3,2,2)^{(+)}$

TABLE III
THE MAPPING $\boldsymbol{\psi}^{(+)}(-1,i)$ FOR EXAMPLE 3.

i	$\boldsymbol{\psi}^{(+)}(-1,i)$
0	$(++--++++----) \Leftrightarrow (2,2,3,4)^{(+)}$
1	$(++---++++---) \Leftrightarrow (2,3,3,3)^{(+)}$
2	$(++----++++--) \Leftrightarrow (2,4,3,2)^{(+)}$
3	$(+++--++----) \Leftrightarrow (3,2,2,4)^{(+)}$
4	$(+++---++---) \Leftrightarrow (3,3,2,3)^{(+)}$
5	$(+++-----++--) \Leftrightarrow (3,4,2,2)^{(+)}$

with $\boldsymbol{\rho}_j = (\rho_{j,1}, \rho_{j,2}, \ldots, \rho_{j,\tilde{n}})$, polarity word $\boldsymbol{\beta}_j = (\beta_{j,1}, \beta_{j,2}, \ldots, \beta_{j,\tilde{n}})$ and

$$m = \sum_{l=1}^{\tilde{n}} \rho_{j,l}.$$

The encoding algorithm for each $\boldsymbol{z}_j$ and the decoding algorithm for each $\boldsymbol{c}_j$ is the same as for VL2 with the appropriate change of $\tilde{m}$ to m and n to $\tilde{n}$. Also, at the decoder, the partitioning of the sequence $\boldsymbol{c} = (c_1, c_2, \ldots, c_{l-1}, c_l, \ldots)$ is into fixed-length sub-sequences $\boldsymbol{c}_j = (\boldsymbol{z}'_j, \boldsymbol{s}_j)$, $j = 1, 2, \ldots$, where the length of $\boldsymbol{c}_j$ is $m + d' + r$ since the lengths of $\boldsymbol{z}'_j$ and $\boldsymbol{s}_j$ are $m + d'$ and r, respectively.

Example 3: Consider the case $d' = 2$, $k' = 4$ and $m = 11$. Since $m + d'$ is odd, the suffix length r must also be odd. Since

$$\max\{\tilde{n}\} = \left\lfloor \frac{m}{d'} \right\rfloor + 1 = \left\lfloor \frac{11}{2} \right\rfloor + 1 = 6,$$

the suffix length is the minimum r that satisfies

$$6 \leq \min\Big\{U^+(2,4,1,r), U^+(2,4,-1,r)\Big\}.$$

Then, as

$$U^+_{(2,4,1)}(z) = z^5 + 2z^7 + 4z^9 + 8z^{11} + 18z^{13} + \ldots$$

and

$$U^+_{(2,4,-1)}(z) = z^5 + z^7 + 2z^9 + 6z^{11} + 11z^{13} + \ldots$$

it follows that the minimum possible suffix length is $r = 11$. Tables II and III gives a possible manifestation of the mappings $\boldsymbol{\psi}^{(+)}$ for $a = 1$ and $a = -1$, respectively (recall that $\boldsymbol{\psi}^{(-)}(a,i) = -\boldsymbol{\psi}^{(+)}(-a,i)$).

Suppose that an encoder of some block RLL code of length 11, which maintains run boundaries at codeword boundaries, produces the $(2,4)$-RLL sequence

$$\boldsymbol{z} = (+++---+++++---+++---+--\ldots),$$

where

$$\boldsymbol{z} \Leftrightarrow \boldsymbol{\rho}^{(\beta_1)} = (3,4,4,2,2,3,2,2,\ldots)^{(+)}.$$

Partition $\boldsymbol{z}$ into sub-sequences of constant length $m = 11$. Then $\boldsymbol{z} = (\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots)$ where $\boldsymbol{z}_1 \Leftrightarrow \boldsymbol{\rho}_1^{(\beta_{1,1})} = (3,4,4)^{(+)}$ and $\boldsymbol{z}_2 \Leftrightarrow \boldsymbol{\rho}_2^{(\beta_{2,1})} = (2,2,3,2,2)^{(-)}$.

Encode $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ as follows. For $\boldsymbol{z}_1$, as can be easily verified, $i = 0, 1, 3$ are near-balancing indices. We will use the first balancing index $i = 0$. The length of the interfix run is $\rho' = d' = 2$, and so $\boldsymbol{\rho}'_1 = (2,3,4,4)$ and $\boldsymbol{\beta}_1'^{[0]} = (-+-+)$. Then, with $\boldsymbol{z}'_1 \Leftrightarrow \boldsymbol{\rho}_1'^{(-)}$, $\sigma(\boldsymbol{z}'_1) = \sigma(\boldsymbol{\beta}_1'^{[0]} \odot \boldsymbol{\rho}'_1) = 1$. Thus, $\boldsymbol{s}_1 = \boldsymbol{\psi}^{(-)}(-1,0) = -\boldsymbol{\psi}^{(+)}(1,0) = (--++----++++) \Leftrightarrow (2,2,4,3)^{(-)} = \boldsymbol{\tau}_1^{(-)}$, and so $\boldsymbol{c}_1 = (\boldsymbol{z}'_1, \boldsymbol{s}_1)$.

For $\boldsymbol{z}_2$, as can be easily verified, $i = 0, 1, 2, 3, 4, 5$ are all near-balancing indices. For illustrative purposes, we will use $i = 2$ as the near-balancing index even though it is not the smallest such index. The length of the interfix run is $\rho' = d' = 2$, and so $\boldsymbol{\rho}'_2 = (2,2,2,3,2,2)$ and $\boldsymbol{\beta}_2'^{[2]} = (+-+-+-)$. Then, with $\boldsymbol{z}'_2 \Leftrightarrow \boldsymbol{\rho}_2'^{(+)}$, $\sigma(\boldsymbol{z}'_2) = \sigma(\boldsymbol{\beta}_2'^{[2]} \odot \boldsymbol{\rho}'_2) = -1$. Thus, $\boldsymbol{s}_2 = \boldsymbol{\psi}^{(+)}(1,2) = (++--++++---) \Leftrightarrow (3,2,3,3)^{(+)} = \boldsymbol{\tau}_2^{(+)}$. However, since $s_{1,11} = +1 = \beta'_{2,1}$, $\boldsymbol{c}_2 = (-\boldsymbol{z}'_2, -\boldsymbol{s}_2) \Leftrightarrow (\boldsymbol{\rho}_2'^{(-)}, \boldsymbol{\tau}_2^{(-)})$.

Therefore, the output of the FL encoder is

$$\boldsymbol{c} = (\boldsymbol{c}_1, \boldsymbol{c}_2) \Leftrightarrow (2,3,4,4,2,2,4,3,2,2,2,3,2,2, 3,2,3,3\ldots)^{(-)}.$$

At the FL decoder, the first 13 symbols are $\boldsymbol{z}'_1$, the next 11 symbols are $\boldsymbol{s}_1$, the next 13 symbols are $\boldsymbol{z}'_2$ and the next 11 symbols are $\boldsymbol{s}_2$. Thus

$$\boldsymbol{z}'_1 \Leftrightarrow \boldsymbol{\rho}_1'^{(-)} = (2,3,4,4)^{(-)}$$
$$\boldsymbol{s}_1 \Leftrightarrow (2,2,4,3)^{(-)}$$

and

$$\boldsymbol{z}'_2 \Leftrightarrow \boldsymbol{\rho}_2'^{(-)} = (2,2,2,3,2,2)^{(-)}$$
$$\boldsymbol{s}_2 \Leftrightarrow (3,2,3,3)^{(-)}.$$

For $\boldsymbol{c}_1 = (\boldsymbol{z}'_1, \boldsymbol{s}_1)$, the near-balancing index $i = 0$ is obtained from $(-1,0) = [\boldsymbol{\psi}^{(-)}]^{-1}(\boldsymbol{s}_1)$ (note that $(1,0) = [\psi^{(+)}]^{-1}(-\boldsymbol{s}_1)$). Then delete the interfix run at index $i+1 = 1$ from $\boldsymbol{\rho}'_1$ to obtain

$$\boldsymbol{z}_1 \Leftrightarrow \boldsymbol{\rho}_1^{(+)} = (3,4,4)^{(+)}.$$

For $\boldsymbol{c}_2 = (\boldsymbol{z}'_2, \boldsymbol{s}_2)$, the near-balancing index $i = 2$ is obtained from $(-1,2) = [\boldsymbol{\psi}^{(-)}]^{-1}(\boldsymbol{s}_2)$ (note that $(1,2) = [\psi^{(+)}]^{-1}(-\boldsymbol{s}_2)$). Then delete the interfix run at index $i+1 = 3$ from $\boldsymbol{\rho}'_2$ and invert the polarity of the first $i = 2$ runs to obtain

$$\boldsymbol{z}_2 \Leftrightarrow \boldsymbol{\rho}_2^{(+)} = (2,2,3,2,2)^{(+)}.$$

Since $z_{1,11} = \beta_{2,1} = +1$, $\boldsymbol{z}_2 \Leftrightarrow \boldsymbol{\rho}_2^{(-)} = (2,2,3,2,2)^{(-)}$. Therefore

$$\boldsymbol{z} = (\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots) \Leftrightarrow (3,4,4,2,2,3,2,2,\ldots)^{(+)},$$

which corresponds to the original RLL sequence. □

B. Balanced RLL Codes with Runlength Constraint Violating Marker

Based on Theorem 1 and the RLL Knuth-like balancing method presented in Section III, we propose a simple and efficient maximum runlength constraint violating marker. The purpose of such a violation is to demarcate the balancing index. The marker, whose first run violates the maximum runlength constraint, is inserted at the index $i+1$, where i is the balancing index. The decoder, by identifying the first runlength violation, then knows the value of the balancing index. The aim is to minimize both the marker length and violation amount, however, as shown in [19], an improvement of one is accompanied by a degradation of the other.

As for VL1 codes, perform the same partitioning of the sequence $\boldsymbol{z} = (z_1, z_2, \ldots, z_{l-1}, z_l, \ldots)$ into variable-length sub-sequences $\boldsymbol{z}_j$, $j = 1, 2, \ldots$, each consisting of n runs, where n is odd, so that

$$\boldsymbol{z} = (\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_{j-1}, \boldsymbol{z}_j, \ldots),$$

where

$$\boldsymbol{z}_j = (z_{j,1}, z_{j,2}, \ldots, z_{j,\tilde{m}}) \Leftrightarrow \boldsymbol{\rho}_j^{(\beta_{j,1})}$$

with $\boldsymbol{\rho}_j = (\rho_{j,1}, \rho_{j,2}, \ldots, \rho_{j,n})$, polarity word $\boldsymbol{\beta}_j = (\beta_{j,1}, \beta_{j,2}, \ldots, \beta_{j,n})$ and

$$\tilde{m} = \sum_{l=1}^{n} \rho_{j,l}.$$

By Theorem 1, for each $\boldsymbol{z}_j$ there exists at least one index i, $0 \le i \le n$, such that $d' \le \sigma(\boldsymbol{\beta}_j^{[i]} \odot \boldsymbol{\rho}_j) \le k'$ (σ_i is a peak in the positive primary region) or $-k' \le \sigma(\boldsymbol{\beta}_j^{[i]} \odot \boldsymbol{\rho}_j) \le -d'$ (σ_i is a valley in the negative primary region). First consider the case where σ_i is a peak. Construct a marker $\boldsymbol{\mu}_j \Leftrightarrow (\rho'_{j,1}, \rho'_{j,2}, \rho'_{j,3})^{(\beta'_{j,1})}$, associated with a polarity word $(\beta'_{j,1}, \beta'_{j,2}, \beta'_{j,3})$, where

$$\begin{aligned}
\rho'_{j,1} &= k' + 1, \\
\rho'_{j,2} &= -\sigma_i + (k' + 1) + d', \\
\rho'_{j,3} &= d',
\end{aligned}$$

and $\beta'_{j,1} = -1$ by (2). Notice that the marker's first run has a violation of 1 (1 greater than k'). Insert the marker at index $i+1$, thereby obtaining

$$\boldsymbol{\rho}''_j \triangleq (\rho_{j,1}, \rho_{j,2}, \ldots, \rho_{j,i}, \rho'_{j,1}, \rho'_{j,2}, \rho'_{j,3}, \rho_{j,i+1}, \ldots, \rho_{j,n})$$

and

$$\boldsymbol{\beta}''^{[i]}_j \triangleq (-\beta_{j,1}, -\beta_{j,2}, \ldots, -\beta_{j,i}, \beta'_{j,1}, \beta'_{j,2}, \beta'_{j,3}, \beta_{j,i+1}, \ldots, \beta_{j,n}).$$

Then, if

$$\boldsymbol{z}''_j \Leftrightarrow \boldsymbol{\rho}''^{(\beta''_{j,1})}_j,$$

where $\beta''_{j,1}$ is the first symbol in $\boldsymbol{\beta}''^{[i]}_j$, it follows that

$$\begin{aligned}
\sigma(\boldsymbol{z}''_j) &= \sigma(\boldsymbol{\beta}''^{[i]}_j \odot \boldsymbol{\rho}''_j) \\
&= \sigma(\boldsymbol{\beta}_j^{[i]} \odot \boldsymbol{\rho}_j) - \rho'_1 + \rho'_2 - \rho'_3 \\
&= \sigma_i - (k' + 1) - \sigma_i + (k' + 1) + d' - d' \\
&= 0.
\end{aligned}$$

Thus, the unbalance of the marker is chosen to be the negative of σ_i, so that, after the insertion of the marker, $\boldsymbol{z}''_j$ is balanced. A moment's reflection reveals that the marker defined above is the optimal achievable using the RLL balancing method: with the first runlength set at $k'+1$, the remaining unbalance is $d' - k' - 1 \le \sigma_i - (k' + 1) \le -1$, since $d' \le \sigma_i \le k'$, which then requires at least two more runs to compensate for the unbalance, the number of runs in the marker must be odd to prevent run mergers and $\rho'_{j,3}$ is set to the minimum possible value, i.e., d'. Furthermore, note that $d' + 1 \le \rho'_{j,2} \le k' + 1$, meaning that $\rho'_{j,2}$ may introduce another 1 violation. $\rho'_{j,2} = k' + 1$ occurs only when $\sigma_i = d'$. Therefore, the marker's maximum violation is 2 (two runs with 1 violation).

If σ_i is a valley, then the marker has

$$\begin{aligned}
\rho'_{j,1} &= k' + 1, \\
\rho'_{j,2} &= \sigma_i + (k' + 1) + d', \\
\rho'_{j,3} &= d',
\end{aligned}$$

and $\beta'_{j,1} = +1$ by (2). Then

$$\begin{aligned}
\sigma(\boldsymbol{z}''_j) &= \sigma(\boldsymbol{\beta}''^{[i]}_j \odot \boldsymbol{\rho}''_j) \\
&= \sigma(\boldsymbol{\beta}_j^{[i]} \odot \boldsymbol{\rho}_j) + \rho'_1 - \rho'_2 + \rho'_3 \\
&= \sigma_i + (k' + 1) - \sigma_i - (k' + 1) - d' + d' \\
&= 0.
\end{aligned}$$

Again, $d' + 1 \le \rho'_{j,2} \le k' + 1$.

If $\beta''_{j,1} = -\beta''_{j-1,n+3}$, where $-\beta''_{j-1,n+3}$ is the last symbol in $\boldsymbol{\beta}''^{[i]}_{j-1}$, send $\boldsymbol{c}_j = \boldsymbol{z}''_j$, else if $\beta''_{j,1} = \beta''_{j-1,n+3}$, send the inverse of $\boldsymbol{z}''_j$, i.e., $\boldsymbol{c}_j = -\boldsymbol{z}''_j \Leftrightarrow \boldsymbol{\rho}''^{(-\beta''_{j,1})}_j$.

Therefore, the output of the encoder corresponding to $\boldsymbol{z}$ is

$$\begin{aligned}
\boldsymbol{c} &= (\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_{j-1}, \boldsymbol{c}_j, \ldots) \\
&= (c_1, c_2, \ldots, c_{l-1}, c_l, \ldots).
\end{aligned}$$

This is also the input to the decoder. The decoder partitions the sequence $\boldsymbol{c} = (c_1, c_2, \ldots, c_{l-1}, c_l, \ldots)$ into sub-sequences $\boldsymbol{z}''_j$, $j = 1, 2, \ldots$, consisting of $n+3$ runs, where

$$\boldsymbol{z}''_j \Leftrightarrow \boldsymbol{\rho}''^{(\beta''_{j,1})}_j,$$

with $\boldsymbol{\rho}''_j = (\rho''_{j,1}, \rho''_{j,2}, \ldots, \rho''_{j,n+3})$ and a polarity word $\boldsymbol{\beta}''_j = (\beta''_{j,1}, \beta''_{j,2}, \ldots, \beta''_{j,n+3})$. Identify the first $\rho''_{j,l} = k' + 1$ in $\boldsymbol{\rho}''_j$, then $i = l - 1$ is the balancing index. Obtain $\boldsymbol{\rho}_j$ from $\boldsymbol{\rho}''_j$ by deleting the marker $(\rho''_{j,i+1}, \rho''_{j,i+2}, \rho''_{j,i+3})$ at index $i+1$, i.e.,

$$\boldsymbol{\rho}_j = (\rho''_{j,1}, \rho''_{j,2}, \ldots, \rho''_{j,i-1}, \rho''_{j,i}, \rho''_{j,i+4}, \ldots, \rho''_{j,n+3}).$$

Also obtain $\boldsymbol{\beta}_j$ from $\boldsymbol{\beta}''_j$ by deleting $(\beta''_{j,i+1}, \beta''_{j,i+2}, \beta''_{j,i+3})$ at index $i+1$ and inverting all $\beta''_{j,l}$ for $l = 1, 2, \ldots, i$ (thereby undoing the inversion introduced during the RLL balancing method), i.e.,

$$\boldsymbol{\beta}_j = (-\beta''_{j,1}, -\beta''_{j,2}, \ldots, -\beta''_{j,i-1}, -\beta''_{j,i}, \beta''_{j,i+4}, \ldots, \beta''_{j,n+3}).$$

Then the original RLL word is $\boldsymbol{z}_j = (z_{j,1}, z_{j,2}, \ldots, z_{j,\tilde{m}}) \Leftrightarrow \boldsymbol{\rho}_j^{(\beta_{j,1})}$, if $\beta_{j,1} = -z_{j-1,\tilde{m}}$, and $\boldsymbol{z}_j \Leftrightarrow \boldsymbol{\rho}_j^{(-\beta_{j,1})}$, if $\beta_{j,1} = z_{j-1,\tilde{m}}$, where $\beta_{j,1}$ is the first symbol in $\boldsymbol{\beta}_j$.

VI. Performance Comparison

Having described the various balanced RLL codes based on the RLL Knuth-like balancing method, in this section we compare the performance of these codes in terms of code rate with existing balanced RLL codes which are based on Knuth's original balancing method. It is demonstrated that the proposed codes outperform their counterparts for corresponding parameters almost universally, and that, in certain cases, the improvement is significant.

A. VL1, VL2 and FL Codes

In evaluating VL1 and VL2 codes, which are variable length, the average length of the variable-length portions are needed for comparison purposes. In case of VL1 and VL2 codes, the length of the (d', k')-RLL source words is variable. If

$$\boldsymbol{z} = (z_1, z_2, \ldots, z_{\tilde{m}}) \Leftrightarrow \boldsymbol{\rho}^{(\beta_1)} = (\rho_1, \rho_2, \ldots, \rho_n)^{(\beta_1)}$$

denotes the RLL source word, the length of $\boldsymbol{z}$,

$$\tilde{m} = \sum_{j=1}^{n} \rho_j,$$

is variable length since n, the number of runs in $\boldsymbol{z}$, is fixed for VL1 and VL2 codes. Let $\bar{m}$ denote the average length of $\tilde{m}$. Then $\bar{m}$ can be computed in the following manner. Assume that the RLL source words $\boldsymbol{z}$ are produced by a maxentropic source. Then, for RLL codes whose code rate is close to capacity, this is a reasonable approximation. For a maxentropic source, $\boldsymbol{\rho}$, which is fixed length, can be any word in $\mathcal{L}^n_{(d',k')}$. This can be equivalently reformulated in the following manner. Let $\mathcal{A} = \{0, 1, \ldots, k' - d'\}$ be a q-ary alphabet with $q = k' - d' + 1$. Let $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n) \in \mathcal{A}^n$ be any word of length n over the alphabet $\mathcal{A}$. Then, any (d', k')-RLL word $\boldsymbol{\rho}$ can be obtained from $\boldsymbol{\alpha}$ as $\boldsymbol{\rho} = (\alpha_1 + d', \alpha_2 + d', \ldots, \alpha_n + d')$. Since we consider all possible $\boldsymbol{\alpha} \in \mathcal{A}^n$ and assume that the occurrence of each $\boldsymbol{\alpha}$ is equiprobable and independent, the probability of each symbol in $\mathcal{A}$ at any index j, $1 \le j \le n$, is $1/q$. Therefore

$$\begin{aligned}
\bar{m} &= n \sum_{j=0}^{q-1} \frac{j + d'}{q} \\
&= \frac{n}{q}\Big(qd' + \sum_{j=0}^{q-1} j\Big) \\
&= nd' + \frac{n}{q}\frac{q(q-1)}{2} \\
&= \frac{n(d' + k')}{2}. \qquad (13)
\end{aligned}$$

While the interfix length $\rho' = d'$ for VL2 and FL codes is fixed, it is variable length $d' \le \rho' \le k'$ for VL1 codes. Since determining the average interfix lengths theoretically is a particularly difficult endeavor, we obtained the average interfix lengths through simulations by generating all variable length (d', k')-RLL words containing n runs and applied the RLL balancing method from Section III. Table IV summarizes the results, where the entry without brackets denotes the average interfix length when the first occurring balancing index is selected, and the entry inside brackets gives the average interfix length when the balancing index with the smallest interfix length is selected. It is interesting to note that the prior entry is always equal to $(d' + k')/2$. Also, for the latter entry, notice that for $k' > d' + 1$ it decreases as n increases. For the VL1 codes, we will use $(d' + k')/2$ as the average interfix length.

TABLE IV
Average interfix lengths for VL1 codes. The entry without brackets denotes the average interfix length when the first occurring balancing index is selected, and the entry inside brackets gives the average interfix length when the balancing index with the smallest interfix length is selected.

	$n =$			
	3	5	7	9
$(d', k') =$				
$(1, 2)$	1.5 (1.5)	1.5 (1.5)	1.5 (1.5)	1.5 (1.5)
$(1, 3)$	2.0 (1.78)	2.0 (1.67)	2.0 (1.62)	2.0 (1.6)
$(1, 4)$	2.5 (2.12)	2.5 (1.93)	2.5 (1.84)	2.5 (1.78)
$(1, 5)$	3.0 (2.47)	3.0 (2.21)	3.0 (2.07)	3.0 (1.98)
$(1, 6)$	3.5 (2.83)	3.5 (2.5)	3.5 (2.32)	3.5 (2.2)
$(1, 7)$	4.0 (3.2)	4.0 (2.79)	4.0 (2.57)	4.0 (2.43)
$(1, 8)$	4.5 (3.56)	4.5 (3.09)	4.5 (2.83)	4.5 (2.67)
$(2, 3)$	2.5 (2.5)	2.5 (2.5)	2.5 (2.5)	2.5 (2.5)
$(2, 4)$	3.0 (2.78)	3.0 (2.67)	3.0 (2.62)	3.0 (2.6)
$(2, 5)$	3.5 (3.12)	3.5 (2.93)	3.5 (2.84)	3.5 (2.78)
$(2, 6)$	4.0 (3.47)	4.0 (3.21)	4.0 (3.07)	4.0 (2.98)
$(2, 7)$	4.5 (3.83)	4.5 (3.5)	4.5 (3.32)	4.5 (3.2)
$(2, 8)$	5.0 (4.2)	5.0 (3.79)	5.0 (3.57)	5.0 (3.43)
$(3, 4)$	3.5 (3.5)	3.5 (3.5)	3.5 (3.5)	3.5 (3.5)
$(3, 5)$	4.0 (3.78)	4.0 (3.67)	4.0 (3.62)	4.0 (3.6)
$(3, 6)$	4.5 (4.12)	4.5 (3.93)	4.5 (3.84)	4.5 (3.78)
$(3, 7)$	5.0 (4.47)	5.0 (4.21)	5.0 (4.07)	5.0 (3.98)
$(3, 8)$	5.5 (4.83)	5.5 (4.5)	5.5 (4.32)	5.5 (4.2)
$(4, 5)$	4.5 (4.5)	4.5 (4.5)	4.5 (4.5)	4.5 (4.5)
$(4, 6)$	5.0 (4.78)	5.0 (4.67)	5.0 (4.62)	5.0 (4.6)
$(4, 7)$	5.5 (5.12)	5.5 (4.93)	5.5 (4.84)	5.5 (4.78)
$(4, 8)$	6.0 (5.47)	6.0 (5.21)	6.0 (5.07)	6.0 (4.98)
$(5, 6)$	5.5 (5.5)	5.5 (5.5)	5.5 (5.5)	5.5 (5.5)
$(5, 7)$	6.0 (5.78)	6.0 (5.67)	6.0 (5.62)	6.0 (5.6)
$(5, 8)$	6.5 (6.12)	6.5 (5.93)	6.5 (5.84)	6.5 (5.78)
$(6, 7)$	6.5 (6.5)	6.5 (6.5)	6.5 (6.5)	6.5 (6.5)
$(6, 8)$	7.0 (6.78)	7.0 (6.67)	7.0 (6.62)	7.0 (6.6)
$(7, 8)$	7.5 (7.5)	7.5 (7.5)	7.5 (7.5)	7.5 (7.5)

Let $\bar{M}_1(d', k', r)$, $\bar{M}_2(d', k', r)$ and $M_3(d', k', r)$ denote the maximum possible RLL source word length for a suffix length r for the VL1, VL2 and FL codes, respectively. For the VL1 codes, the number of runs in the RLL source word is n, where n is odd, and

$$n \le B(d', k', r).$$

If $B(d', k', r)$ is odd, then by (13)

$$\bar{M}_1(d', k', r) = \frac{d' + k'}{2} B(d', k', r).$$

Otherwise, if $B(d',k',r)$ is even, then

$$\bar{M}_1(d',k',r) = \frac{d'+k'}{2}[B(d',k',r)-1].$$

For a RLL random walk (i,σ_i), $0 \le i \le n$, $\sigma_l - \sigma_{l-1} \in 2\mathcal{L}_{(d',k')}$ for $1 \le l \le n$. Therefore, σ_i is always even (odd) if σ_0 is even (odd). For the VL2 codes, the RLL source word contains n runs, where n is odd. In such a case, σ_0 can be even or odd since the length of $\boldsymbol{z}_j$, $\tilde{m} = \sum_{j=1}^{n} \rho_j$, can be even or odd. The unbalance of $\boldsymbol{z}'_j$ is $\sigma_i - d'$ if σ_i is a peak, and $\sigma_i + d'$ if σ_i is a valley. Therefore, the unbalance of $\boldsymbol{z}'_j$ can be even or odd. Since the suffix needs to compensate for this unbalance, suffixes of both even and odd length are required.

Let

$$U_{\min}(d',k',r) \triangleq \min\Big\{U(d',k',a,r) : a \in \mathcal{L}_{(-k'+d',k'-d')}, \; a \text{ is even}\Big\},$$

if r is even, and

$$U_{\min}(d',k',r) \triangleq \min\Big\{U(d',k',a,r) : a \in \mathcal{L}_{(-k'+d',k'-d')}, \; a \text{ is odd}\Big\},$$

if r is odd. Since

$$n \le U_{\min}(d',k',r),$$

then

$$\bar{M}_2(d',k',r) = \frac{d'+k'}{2} U_{\min}(d',k',r),$$

if $U_{\min}(d',k',r)$ is odd, and

$$\bar{M}_2(d',k',r) = \frac{d'+k'}{2}[U_{\min}(d',k',r)-1],$$

if $U_{\min}(d',k',r)$ is even.

For the FL codes,

$$\left\lfloor \frac{m}{d'} \right\rfloor + 1 \le U_{\min}(d',k',r),$$

and so the maximum possible m is

$$\begin{aligned} m &= d'[U_{\min}(d',k',r)-1] + d' - 1 \\ &= d' U_{\min}(d',k',r) - 1. \end{aligned}$$

If m is even (odd) and r is even (odd), then

$$M_3(d',k',r) = m,$$

else if m is odd (even) and r is even (odd), then

$$M_3(d',k',r) = m-1.$$

For the balanced RLL code Construction 3 from [7], where $0 < 2d \le k < \infty$,

$$N_3(d,k,r) = \min\Big\{\hat{U}(d+1,k+1,a,r) : a \in \{-d-1,-d+1,\ldots,d-1,d+1\}\Big\},$$

where $N_3(d,k,r)$ is the maximum possible RLL source word length for a suffix of length r.

While Construction 3 from [7] guarantees the preservation of the runlength constraints within the balanced RLL codeword that it generates, there is no mechanism to ensure that such runlength constraints are preserved at boundaries of such codewords. This limitation means that codewords generated by Construction 3 cannot be concatenated freely without potentially introducing runlength violations. Note that suffixes of Construction 3 have a mechanism to deal with case where the last run in the RLL source word is less than d' [7, p. 320], implying that RLL source word boundaries need not correspond to run boundaries. Furthermore, a suffix for Construction 3 may end with a run whose length is between 1 and k', both inclusive (cf. the definition of $\hat{U}^+(d',k',a,r)$), and so a runlength violation may occur at the boundary between the suffix of the current codeword and the next codeword. The codes VL1, VL2 and FL do not have this limitation, i.e., their codewords can be concatenated freely without any runlength violations. As a result, a direct comparison of $\bar{M}_1$, $\bar{M}_2$ and M_3 with N_3 is somewhat skewed. However, a simple modification to Construction 3 from [7] can remove this limitation.

Consider the case where a run spans the boundary of two (d,k)-constrained source words, i.e., one part of the run is in the first source word while the other part is in the second source word. Between these two source words is inserted a (d,k)-constrained suffix $\boldsymbol{f}^{-1}(\boldsymbol{s})$, where $\boldsymbol{s} \in \mathcal{U}^{(-)}(d+1,k+1,a,r)$, i.e., $\boldsymbol{f}^{-1}(\boldsymbol{s})$ starts with 1 and ends with at least d and at most k 0s.

Consider two runs with lengths ρ_1, $d' \le \rho_1 \le k'$, and ρ_2, $d' \le \rho_2 \le k'$. Partition ρ_1 into two non-zero portions of lengths $\rho_{1,1}$ and $\rho_{1,2}$ where $\rho_1 = \rho_{1,1} + \rho_{1,2}$. Between these two portions of the split run of length ρ_1 is inserted a run of length ρ_2, which is also split into two portions of lengths $\rho_{2,1}$ and $\rho_{2,2}$ such that $\rho_2 = \rho_{2,1} + \rho_{2,2}$. Then $\rho_{1,1}$ and $\rho_{2,1}$ are combined into a single run of length $\rho_{1,1} + \rho_{2,1}$, as are $\rho_{2,2}$ and $\rho_{1,2}$ into a run of length $\rho_{2,2} + \rho_{1,2}$. It is desired that the resulting two new runs satisfy the runlength constraints, i.e., $d' \le \rho_{1,1} + \rho_{2,1} \le k'$ and $d' \le \rho_{2,2} + \rho_{1,2} \le k'$. The partitioning of ρ_1 is predetermined, while ρ_2 can be partitioned in any way. Consider the following cases:

1) $\rho_{1,1} < d'$: Set $\rho_{2,2} = \rho_{1,1}$, therefore $\rho_{2,2} + \rho_{1,2} = \rho_{1,1} + \rho_{1,2} = \rho_1$ and so $d' \le \rho_{2,2} + \rho_{1,2} \le k'$. Then $\rho_{2,1} = \rho_2 - \rho_{2,2} = \rho_2 - \rho_{1,1}$, and therefore $d' \le \rho_{1,1} + \rho_{2,1} \le k'$.
2) $\rho_{1,2} < d'$: Set $\rho_{2,1} = \rho_{1,2}$, therefore $\rho_{1,1} + \rho_{2,1} = \rho_{1,1} + \rho_{1,2} = \rho_1$ and so $d' \le \rho_{1,1} + \rho_{2,1} \le k'$. Then $\rho_{2,2} = \rho_2 - \rho_{2,1} = \rho_2 - \rho_{1,2}$, and therefore $d' \le \rho_{1,2} + \rho_{2,2} \le k'$.
3) $\rho_{1,1} \ge d'$ and $\rho_{1,2} \ge d'$: Set $\rho_{2,1} = \min\{\rho_{1,2}, \rho_2 - 1\}$. If $\rho_{1,2} \le \rho_2 - 1$, then $\rho_{2,1} = \rho_{1,2}$ and this is the same as case 2) above. If $\rho_2 - 1 < \rho_{1,2}$, then it follows that $d' \le \rho_{1,1} + \rho_{2,1} = \rho_{1,1} + \rho_2 - 1 < \rho_{1,1} + \rho_{1,2} = \rho_1 \le k'$. Then $\rho_{2,2} = 1$, and $d' \le \rho_{2,2} + \rho_{1,2} = 1 + \rho_{1,2} \le \rho_{1,1} + \rho_{1,2} = \rho_1 \le k'$.

If ρ_1 corresponds to the length of the run that is split over two RLL source words and ρ_2 corresponds to the length of the last run in $\boldsymbol{s}$, then by appropriately cyclically shifting the suffix $\boldsymbol{s}$, the runlength constraint across the suffix and the next balanced RLL codeword can be preserved. For a word $\boldsymbol{x} = (x_1, x_2, \ldots, x_r)$, denote the cyclic shift of $a > 0$ symbols to the right as $\boldsymbol{x}^{\langle a \rangle} \triangleq (x_{r-a+1}, \ldots, x_r, x_1, x_2, \ldots, x_{r-a})$. If $a < 0$, then cyclically shift $|a|$ symbols to the left, i.e., for

$a > 0$, $\boldsymbol{x}^{\langle -a \rangle} \triangleq (x_{a+1}, \ldots, x_r, x_1, x_2, \ldots, x_a)$. Then

1) If $\rho_{1,1} < d'$: use $\boldsymbol{f}^{-1}(\boldsymbol{s})^{\langle \rho_2 - \rho_{1,1} \rangle}$ as the suffix.
2) If $\rho_{1,2} < d'$: use $\boldsymbol{f}^{-1}(\boldsymbol{s})^{\langle \rho_{1,2} \rangle}$ as the suffix.
3) If $\rho_{1,1} \geq d'$ and $\rho_{1,2} \geq d'$: use $\boldsymbol{f}^{-1}(\boldsymbol{s})^{\langle \min\{\rho_{1,2}, \rho_2 - 1\} \rangle}$ as the suffix.

The cyclically shifted version of the suffix, $\boldsymbol{s}^{\langle a \rangle}$, must have the same unbalance as the original, unshifted suffix, i.e., $\sigma(\boldsymbol{s}^{\langle a \rangle}) = \sigma(\boldsymbol{s})$. This will be the case if the polarity of the first run is the same as the last run in $\boldsymbol{s}^{\langle a \rangle}$ after shifting. This happens only if the number of runs in $\boldsymbol{s}$ is even.

Let $\hat{\boldsymbol{s}} = \boldsymbol{f}^{-1}(\boldsymbol{s})^{\langle a \rangle}$, where $a = 0$ or $a = \rho_2 - \rho_{1,1}$ or $a = \rho_{1,2}$ or $a = \rho_2 - 1$, denote the selected suffix. The decoder, which can extract $\hat{\boldsymbol{s}}$, then locates the index j of the first symbol 1 in $\hat{\boldsymbol{s}}$. Then

1) If $j = 1$: the original suffix is $\hat{\boldsymbol{s}}$.
2) If $j > 1$: the original suffix is $\hat{\boldsymbol{s}}^{\langle -j+1 \rangle}$.

The balancing index can then be determined from the original suffix.

Let $U'^{+}(d', k', a, r)$ ($U'^{-}(d', k', a, r)$) denote the number of (d', k')-RLL words of length r that start with '+1' ('−1'), have an unbalance of a and consist of an even number of runs. Clearly, $U'^{+}(d', k', a, r) = U'^{-}(d', k', -a, r)$. Let

$$U'(d', k', a, r) \triangleq U'^{+}(d', k', a, r) = U'^{-}(d', k', -a, r).$$

Then,

$$U'^{+}(d', k', a, r) = [z^r] U'^{+}_{(d', k', a)}(z),$$

where

$$U'^{+}_{(d', k', a)}(z) = U'^{+}_{(d', k', a)}(z^2, 1) z^{-|a|}$$

and

$$U'^{+}_{(d', k', a)}(z, u) = R_{(d', k')}(z, u) \odot R_{(d', k')}(z, u) z^{|a|}.$$

With this modification to Construction 3 from [7], which allows balanced RLL codewords to be concatenated freely without any runlength constraint violations, the maximum RLL source word length $N'_3(d, k, r)$ for a suffix of length r is

$$N'_3(d, k, r) = \min\Big\{ U'(d+1, k+1, a, r) : a \in \{-d-1, -d+1, \ldots, d-1, d+1\} \Big\}.$$

Finally, for Construction 4 from [7], where $d = 0$ and which is a special case of Construction 3, the maximum RLL source word length for a suffix length r is denoted by

$$N_4(k, r) = \min\Big\{ U(1, k+1, -1, r), U(1, k+1, 1, r) \Big\}.$$

Let r' denote the total redundancy of the various balanced RLL codes to be compared. The total redundancy is the sum of the interfix and suffix lengths. Then $r' = r + (d' + k')/2$ for the VL1 codes, while $r' = r + d'$ for the VL2 and FL codes. For Construction 3 from [7], $r' = r + d + 1$, and for Construction 4 from [7], $r' = r + 1$. We are going to compare the maximum RLL source word length that can be achieved by the various codes at a specific total redundancy r'. In all the subsequent tables, VL1 codes are evaluated as $\bar{M}_1(d', k', r' - \lceil (d' + k')/2 \rceil)$, where $\lceil x \rceil$ represents the smallest integer greater than or equal to x, VL2 codes as $\bar{M}_2(d', k', r' - d')$, FL codes as $M_3(d', k', r - d')$, Construction 3 codes from [7] as $N_3(d, k, r' - d - 1)$, the modified version of Construction 3 codes as $N'_3(d, k, r' - d - 1)$ and Construction 4 codes from [7] as $N_4(k, r' - 1)$.

TABLE V
MAXIMUM RLL SOURCE WORD LENGTH FOR REDUNDANCY r', $d' = 2, k' = 4$. $\bar{M}_1$, $\bar{M}_2$ AND M_3 DENOTE THE MAXIMUM POSSIBLE RLL SOURCE WORD LENGTH FOR VL1, VL2 AND FL CODES, RESPECTIVELY. N_3 AND N'_3 DENOTE THE MAXIMUM POSSIBLE RLL SOURCE WORD LENGTH FOR CONSTRUCTION 3 FROM [7] AND MODIFIED VERSION OF CONSTRUCTION 3 AS PRESENTED IN THIS ARTICLE, RESPECTIVELY.

r'	$\bar{M}_1$	$\bar{M}_2$	M_3	N_3	N'_3
20	-	153	102	71	46
21	279	315	211	-	-
22	-	291	192	146	85
23	621	687	457	-	-
24	-	717	478	341	211
25	1251	1479	985	-	-
26	-	1455	970	720	406
27	2721	3087	2059	-	-
28	-	3243	2160	1575	886
29	5553	6615	4411	-	-
30	-	6957	4638	3418	1894
31	11949	14079	9385	-	-
32	-	15099	10064	7361	3953
33	24807	29775	19851	-	-
34	-	32253	21500	15828	8369
35	52659	63477	42317	-	-
36	-	70101	46734	34167	17909
37	110601	134805	89871	-	-
38	-	149589	99726	73115	37404
39	233913	286161	190773	-	-
40	-	322653	215102	157108	79803
41	492681	608841	405895	-	-
42	-	691623	461082	336573	168690
43	1042587	1293357	862237	-	-
44	-	1484163	989440	720559	356568

For the first comparison, consider the case $d' = 2$ ($d = 1$) and $k' = 4$ ($k = 3$), with the maximum RLL source word lengths for various codes shown in Table V (cf. [7, Tab. IV]). Note that N'_3 is approximately half of N_3; this is the cost of ensuring that runlength constraints are maintained at codeword boundaries. Notice that $\bar{M}_1$, $\bar{M}_2$ and M_3 all improve on N_3. In the case of $\bar{M}_1$, it does so at a lower r'. For VL2, we have to take the minimum of $\bar{M}_2$ at r' and $r' + 1$ (recall that VL2 requires both even and odd length suffixes), with the average redundancy being $r' + 0.5$ (assuming even and odd suffixes are equiprobable). It is clear that the VL1 codes have a better code rate than FL codes. Also, notice that $\bar{M}_2$ is slightly higher than $\bar{M}_1$, but this is at a redundancy 0.5 higher.

Table VI shows the results for $d' = 2$ ($d = 1$) and $k' = 10$ ($k = 9$). N_3 is larger than $\bar{M}_1$, $\bar{M}_2$ and M_3, except for $\bar{M}_2$ when $r' \geq 42$ where it is larger at a lower redundancy. However, $\bar{M}_1$, $\bar{M}_2$ and M_3 are larger than N'_3 for a sufficiently high r' ($\bar{M}_1$ for all r' in the table).

Table VII shows the results for $d' = 4$ ($d = 3$) and $k' = 8$ ($k = 7$). In this case, the performance improvement is the

TABLE VI
MAXIMUM RLL SOURCE WORD LENGTH FOR REDUNDANCY r', $d' = 2, k' = 10$. $\bar{M}_1$, $\bar{M}_2$ AND M_3 DENOTE THE MAXIMUM POSSIBLE RLL SOURCE WORD LENGTH FOR VL1, VL2 AND FL CODES, RESPECTIVELY. N_3 AND N_3' DENOTE THE MAXIMUM POSSIBLE RLL SOURCE WORD LENGTH FOR CONSTRUCTION 3 FROM [7] AND MODIFIED VERSION OF CONSTRUCTION 3 AS PRESENTED IN THIS ARTICLE, RESPECTIVELY.

r'	$\bar{M}_1$	$\bar{M}_2$	M_3	N_3	N_3'
20	246	90	30	323	136
21	-	402	135	-	-
22	618	438	144	824	338
23	-	1098	367	-	-
24	1554	1182	394	2122	848
25	-	3294	1099	-	-
26	3882	3714	1236	5398	2115
27	-	9222	3075	-	-
28	9702	10602	3534	13818	5334
29	-	25758	8587	-	-
30	24522	30330	10110	35246	13386
31	-	70878	23625	-	-
32	61386	85002	28334	90033	33834
33	-	193230	64411	-	-
34	155622	235686	78560	229886	85344
35	-	522906	174301	-	-
36	391926	647214	215738	587359	216120
37	-	1406430	468811	-	-
38	993690	1764006	588000	1500807	547244
39	-	3763482	1254495	-	-
40	2515650	4775778	1591926	3836857	1388409
41	-	10030278	3343427	-	-
42	6384630	12867150	4289048	9811684	3525249
43	-	26633766	8877923	-	-
44	16213278	34496730	11498910	25101181	8960996

TABLE VII
MAXIMUM RLL SOURCE WORD LENGTH FOR REDUNDANCY r', $d' = 4, k' = 8$. $\bar{M}_1$, $\bar{M}_2$ AND M_3 DENOTE THE MAXIMUM POSSIBLE RLL SOURCE WORD LENGTH FOR VL1, VL2 AND FL CODES, RESPECTIVELY. N_3 AND N_3' DENOTE THE MAXIMUM POSSIBLE RLL SOURCE WORD LENGTH FOR CONSTRUCTION 3 FROM [7] AND MODIFIED VERSION OF CONSTRUCTION 3 AS PRESENTED IN THIS ARTICLE, RESPECTIVELY.

r'	$\bar{M}_1$	$\bar{M}_2$	M_3	N_3	N_3'
20	6	-	-	-	-
21	-	-	-	-	-
22	18	-	-	-	-
23	-	18	15	-	-
24	18	30	18	5	5
25	-	54	39	-	-
26	54	42	30	12	8
27	-	66	47	-	-
28	90	54	34	15	9
29	-	66	47	-	-
30	186	42	30	14	8
31	-	126	87	-	-
32	222	126	82	33	20
33	-	306	207	-	-
34	378	318	214	90	54
35	-	642	431	-	-
36	738	678	454	186	114
37	-	1134	759	-	-
38	1542	1074	718	295	180
39	-	1842	1231	-	-
40	2466	1650	1098	465	260
41	-	2814	1879	-	-
42	3906	2562	1710	808	388
43	-	4854	3239	-	-
44	6498	5118	3414	1585	794

most distinct. $\bar{M}_1$, $\bar{M}_2$ and M_3 are significantly larger than N_3 and N_3' are all r'. For $r' = 44$, $\bar{M}_1$ is approximately eight times N_3'. The VL1 codes have the best code rate, with $\bar{M}_1$ being nearly double M_3 for larger r'.

The final comparison is shown in Table VIII, where $d' = 1$ ($d = 0$) and $k' = 4$ ($k = 3$) (cf. [7, Tab. V]). This corresponds to the case where there is no minimum runlength constraint. This is where it is expected for the VL1, VL2 and FL codes to perform the worst in comparison to the codes from [7] because the RLL Knuth-like balancing method on which these codes are based approaches Knuth's original balancing method when there is no minimum runlength constraint. This is corroborated by Table VIII, where N_4 is higher than M_3. However, the VL2 codes have a better code rate since $\bar{M}_2$ is slightly larger than N_4 at a slightly lower redundancy.

In conclusion, at least one of the codes VL1, VL2 or FL provide better performance than the codes from [7] for all the values of d' and k' considered (where this statement takes into account N_3' rather than N_3 for fairness of comparison). In the majority of cases, all three codes provide better performance. For low values of d', VL2 codes have the best performance, however, their improvement over codes from [7] is relatively marginal. For larger values of d', the VL1 codes have the best performance, and significantly improve on the codes from [7]. In general, the relative improvement increases as d' increases and/or $k' - d'$ decreases. Note that these improvements are obtained with VL1 and VL2 codes, which have no constraints on d' and k', unlike Construction 3 from [7] which has the restriction $k \geq 2d$. A disadvantage of the VL1, VL2 and FL codes is that they are unsuitable when $k' = \infty$, i.e., when there is no maximum runlength constraint. This is because the RLL balancing method that these codes utilize is based on the premise that k' is finite.

B. Marker Codes

As shown in Section V-B, the marker based on the RLL balancing method has a worst case length of $2(k'+1)+d' = 2(k+2)+(d+1) = 2k+d+5$. This is $3d+1$ less than the length of the optimal 1 violation marker from [19] based on Knuth's original balancing method. The advantages of the marker proposed in this paper are:

1) The marker is shorter by $3d+1$ than the optimal marker length with 1 violation based on Knuth's original balancing method [19].
2) The marker has no restriction on d' and k' (optimal markers from [19] have the restriction $k \geq 2d$).

TABLE VIII
MAXIMUM RLL SOURCE WORD LENGTH FOR REDUNDANCY r', $d' = 1, k' = 4$. $\bar{M}_1$, $\bar{M}_2$ AND M_3 DENOTE THE MAXIMUM POSSIBLE RLL SOURCE WORD LENGTH FOR VL1, VL2 AND FL CODES, RESPECTIVELY. N_4 DENOTES THE MAXIMUM POSSIBLE RLL SOURCE WORD LENGTH FOR CONSTRUCTION 4 FROM [7].

r'	$\bar{M}_1$	$\bar{M}_2$	M_3	N_4
10	-	47.5	17	52
11	87.5	167.5	66	-
12	-	197.5	79	185
13	302.5	637.5	254	-
14	-	797.5	319	662
15	1047.5	2372.5	948	-
16	-	3112.5	1245	2372
17	3672.5	8762.5	3504	-
18	-	11942.5	4775	8510
19	12947.5	32212.5	12884	-
20	-	45302.5	18119	30578
21	45872.5	118152.5	47260	-
22	-	170537.5	68213	110069
23	163177.5	432747.5	173098	-
24	-	638447.5	255377	396877
25	582432.5	1583622.5	633448	-
26	-	2380607.5	952243	1433229
27	2084997.5	5792437.5	2316974	-
28	-	8850362.5	3540143	5183060
29	7483022.5	21182412.5	8472964	-
30	-	32829382.5	13131753	18767985

3) Simpler encoding and decoding procedures. Since the runlength boundaries correspond to marker boundaries, the boundary merging (runlength constraint preserving) operations of [19] are unnecessary.
4) A more computationally efficient balancing method since each symbol need not be inverted, but only runs as a whole.

These advantages are achieved at the cost of an additional 1 violation which occurs infrequently.

VII. CONCLUSION

A novel Knuth-like balancing method for RLL words was presented. The method is better suited for RLL words since it inverts runs as a whole, rather than inverting individual symbols as with Knuth's original balancing method. It was shown that under certain conditions, such a balancing procedure is guaranteed to produce at least one balancing or near-balancing point for any RLL source word, which is a prerequisite to constructing balanced RLL codes based on such a balancing method.

Numerous balanced RLL codes were constructed with the proposed Knuth-like balancing method forming the core of each. In the context of traditional balanced RLL codes which strictly adhere to the runlength constraints, three related codes were presented, two variable length and one fixed length. While the variable length codes have a better code rate than the fixed-length codes for all d' and k', the fixed-length version does have the advantage of limiting error propagation. All three codes have better codes rates than existing balanced RLL codes based on Knuth's original balancing method for the majority of d' and k' values. Notably, at least one of the three new codes has better code rate for all d' and k'. The improvement in code rate is significant for the majority of d' and k' values, and is particularly pronounced for larger d' and smaller $k' - d'$. Furthermore, a maximum runlength violating marker based on the new RLL balancing method was proposed that improves on the optimal 1 violation markers based on Knuth's original balancing method in a number of ways. These improvements, including marker length and encoding and decoding complexity, are achieved at the cost of an additional 1 violation which happens infrequently.

Through these various code constructions and associated performance evaluations, we have demonstrated the greater suitability of the novel Knuth-like balancing method to balancing RLL words than Knuth's original balancing method.

REFERENCES

[1] K. A. S. Immink, *Codes for Mass Data Storage Systems*, 2nd ed. Eindhoven, The Netherlands: Shannon Foundation Publishers, 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. Abdel-Ghaffar and J. H. Weber, "Constructing efficient DC-free runlength-limited block codes for recording channels," *IEEE Trans. Inf. Theory*, vol. 46, no. 4, pp. 1599–1602, Jul. 2000.
[5] K. A. S. Immink, J.-Y. Kim, S.-W. Suh, and S. K. Ahn, "Extremely efficient DC-free RLL codes for optical recording," *IEEE Trans. Consum. Electron.*, vol. 47, no. 4, pp. 910–914, Nov. 2001.
[6] V. Braun and K. A. S. Immink, "An enumerative coding technique for DC-free runlength-limited sequences," *IEEE Trans. Commun.*, vol. 48, no. 12, pp. 2024–2031, Dec. 2000.
[7] K. A. S. Immink, J. H. Weber, and H. C. Ferreira, "Balanced runlength limited codes using Knuth's algorithm," in *Proc. of the 2011 IEEE Int. Symp. on Inf. Theory*, St. Petersburg, Russia, Jul. 31–Aug. 5, 2011, pp. 317–320.
[8] D. E. Knuth, "Efficient balanced codes," *IEEE Trans. Inf. Theory*, vol. IT-32, no. 1, pp. 51–53, Jan. 1986.
[9] N. Alon, E. E. Bergmann, D. Coppersmith, and A. M. Odlyzko, "Balancing sets of vectors," *IEEE Trans. Inf. Theory*, vol. 34, no. 1, pp. 128–130, Jan. 1988.
[10] S. Al-Bassam and B. Bose, "On balanced codes," *IEEE Trans. Inf. Theory*, vol. 36, no. 2, pp. 406–408, Mar. 1990.
[11] ——, "Design of efficient balanced codes," *IEEE Trans. Comput.*, vol. 43, no. 3, pp. 362–365, Mar. 1994.
[12] K. A. S. Immink and J. H. Weber, "Very efficient balanced codes," *IEEE J. Sel. Areas Commun.*, vol. 28, no. 2, pp. 188–192, Feb. 2010.
[13] J. H. Weber and K. A. S. Immink, "Knuth's balanced codes revisited," *IEEE Trans. Inf. Theory*, vol. 56, no. 4, pp. 1673–1679, Apr. 2010.
[14] 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.
[15] K. Norris and D. S. Bloomberg, "Channel capacity of charge-constrained run-length-limited codes," *IEEE Trans. Magn.*, vol. MAG-17, no. 6, pp. 3452–3455, Nov. 1981.
[16] A. Manada and H. Morita, "On the capacities of balanced codes with run-length constraints," in *Proc. of the 2017 IEEE Int. Symp. on Inf. Theory*, Aachen, Germany, Jun. 25–30, 2017, pp. 1391–1395.
[17] F. Palunčić and B. T. Maharaj, "Using bivariate generating functions to count the number of balanced runlength-limited words," in *GLOBECOM 2017*, Singapore, Dec. 4–8, 2017.
[18] H. C. Ferreira, J. H. Weber, C. H. Heymann, and K. A. S. Immink, "Markers to construct DC free (d, k) constrained balanced block codes using Knuth's inversion," *Electronics Letters*, vol. 48, no. 19, pp. 1209–1211, Sep. 2012.
[19] J. H. Weber, C. H. Heymann, H. C. Ferreira, and K. A. S. Abdel-Ghaffar, "Optimized markers for balancing runlength-limited sequences in optical recording," in *Proc. of the 2014 IEEE Int. Symp. on Inf. Theory*, Honolulu, USA, Jun. 29–Jul. 4, 2014, pp. 1658–1662.

[20] P. Flajolet and R. Sedgewick, *Analytic Combinatorics*. Cambridge, UK: Cambridge University Press, 2009.

Filip Palunčić 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 4 years in industry as a research and development engineer at IDX, a company specializing in industrial communications. During 2016–2017, he was a post-doctoral research fellow at the Sentech group in Broadband Wireless Multimedia Communications, Department of Electrical, Electronic and Computer Engineering, University of Pretoria. He is currently a faculty member in the Department of Electrical, Electronic and Computer Engineering, University of Pretoria.

His research interests include coding techniques (in particular error control coding and constrained coding), information theory, cognitive radio networks and wireless communications.

B. T. Maharaj received his Ph.D. in Engineering in the area of Wireless Communications from the University of Pretoria. He is a full Professor and Dean and currently holds the research position of Sentech Chair in Broadband Wireless Multimedia Communications in the Department of Electrical, Electronic and Computer Engineering at the University of Pretoria. His research interests are in OFDM-MIMO and massive MIMO systems, cognitive radio resource allocation and 5G cognitive radio sensor networks.

Hendrik C. Ferreira was born and educated in South Africa. He received the D.Sc. (Eng.) degree from the University of Pretoria, Pretoria, South Africa, in 1980.

In 1983, he joined Rand Afrikaans University, Johannesburg, South Africa, where he was promoted to a Professor in 1989. From 1994 to 1999, he served two terms as a Chairman for the Department of Electrical and Electronic Engineering, University of Johannesburg, where he is currently a Chair of the UJ Centre for Telecommunications. His current research interests include digital communications (especially power-line communications) and information theory (especially coding techniques).

Dr. Ferreira was a Chairman of the *Communications and Signal Processing Chapter of the IEEE South Africa section* and the Founding Chairman of the *Information Theory Chapter*. Between 1997 and 2005, he has been the Editor-in-Chief of the TRANSACTIONS OF THE SOUTH AFRICAN INSTITUTE OF ELECTRICAL ENGINEERS. He has served as a Chairman for several conferences, including the 17th IEEE International Symposium on Power-Line Communications and its Applications (ISPLC 2013) that was held in Johannesburg, South Africa.