Optimal Almost-Balanced Sequences

Daniella Bar-Lev*, **Adir Kobovich**†, **Orian Leitersdorf**†, and **Eitan Yaakobi***
*Faculty of Computer Science, Technion – Israel Institute of Technology, Haifa 3200003, Israel
†Faculty of Electrical and Computer Engineering, Technion – Israel Institute of Technology, Haifa 3200003, Israel
{daniellalev, yaakobi}@cs.technion.ac.il, {adir.k, orianl}@campus.technion.ac.il

***Abstract*—This paper presents a novel approach to address the constrained coding challenge of generating *almost-balanced sequences*. While strictly balanced sequences have been well studied in the past, the problem of designing efficient algorithms with small redundancy, preferably constant or even a single bit, for almost balanced sequences has remained unsolved. A sequence is $\varepsilon(n)$-*almost balanced* if its Hamming weight is between $0.5n \pm \varepsilon(n)$. It is known that for any algorithm with a constant number of bits, $\varepsilon(n)$ has to be in the order of $\Theta(\sqrt{n})$, with $\mathcal{O}(n)$ average time complexity. However, prior solutions with a single redundancy bit required $\varepsilon(n)$ to be a linear shift from $n/2$. Employing an iterative method and arithmetic coding, our emphasis lies in constructing almost balanced codes with a single redundancy bit. Notably, our method surpasses previous approaches by achieving the *optimal* balanced order of $\Theta(\sqrt{n})$. Additionally, we extend our method to the non-binary case, considering q-ary almost polarity-balanced sequences for even q, and almost symbol-balanced for $q = 4$. Our work marks the first asymptotically optimal solutions for almost-balanced sequences, for both, binary and non-binary alphabet.**

I. Introduction

Constrained codes have a long history in information theory, with applications to data storage and transmission. In the broadest setting, raw data in such applications is encoded (in a one-to-one manner) into a set of words $\mathcal{S}$ over some alphabet Σ that satisfy prescribed rules. Some rules are imposed due to physical limitations, such as those dictated by energy compliance or by memory cell wear, and are typically translated into cost constraints. Others are imposed as a preventive measure to keep the storage device in a sufficiently-reliable operation region. A celebrated result in constrained coding theory by Knuth has analyzed strictly balanced binary sequences or sequences with a fixed weight [1]. We consider in this work the *almost-balanced* constraint which generalizes the well-known balanced Knuth codes [1] by requiring that the entire message possess a Hamming weight of *approximately* $n/2$.

One motivating application of this work is DNA storage, where *almost balanced GC content* is necessary [2]. During the storage phase in DNA strands, media degradation, and in particular breaks, can arise in DNA due to factors that include radiation, humidity, and high temperatures. In [3], the authors proposed to encapsulate the stored DNA in a silica substrate and then to employ custom error-correcting codes to mitigate the effects of these errors. Another approach to dealing with media degradation is to generate strands of DNA that have approximately balanced GC-content, and this approach has been leveraged in several existing works such as [4]–[6].

The research was funded by the European Union (ERC, DNAStorage, 865630). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them. This work was also supported in part by NSF Grant CCF2212437.

The construction of efficient balanced codes has been extensively studied; see e.g. [1], [7]–[10], and extensions to non-binary balanced codes have been considered in [11]–[15]. Codes that combine the balanced property with certain other constraints, such as run-length limitations, have also been addressed for example in [16]. However, the problem of almost balanced sequences with Hamming weight between $0.5n \pm \varepsilon(n)$ has received a little attention. Under this framework, the goal is to find the optimal number of redundant bits as a function of $\varepsilon(n)$, where $\varepsilon(n)$ can be a function of n, e.g. linear in n, $\log n$, or a constant. No less important is the design of such algorithms.

While Knuth's algorithm is an efficient scheme to strictly balance an arbitrary sequence with $\log n + o(\log n)$ redundancy bits, to design an efficient encoder and decoder with less redundancy or even only a single bit is a non-trivial task. The best known construction that uses a single redundancy bit required $\varepsilon(n)$ to be linear with n [17], while a lower bound asserts that the order of $\varepsilon(n)$ has to be $\Omega(\sqrt{n})$. In this work, we close on this gap and present an explicit encoder that uses a single redundancy bit to balance binary sequences for $\varepsilon(n) = \Theta(\sqrt{n})$, with $\mathcal{O}(n)$ average time complexity.

The rest of this paper is organized as follows. In Section II we introduce the definitions that will be utilized throughout the paper and present the arithmetic coding method. Our construction for binary almost-balanced sequences is presented in Section III and generalizations for almost polarity-balanced and almost symbol-balanced for non-binary alphabet are presented in Section IV. Section V concludes this paper.

II. Definitions, Related Works, and Arithmetic Coding

A. Definitions

Let $\Sigma_q = \{0, 1, \ldots, q-1\}$ be the alphabet of size q and let Σ_q^n be the set of all sequences of length n over Σ_q. The Hamming weight of a sequence $\boldsymbol{x} \in \Sigma_q^n$, denoted by $w(\boldsymbol{x})$, is the number of non-zero symbols in $\boldsymbol{x}$. The concatenation of two sequences $\boldsymbol{x}$ and $\boldsymbol{y}$ is denoted by $\boldsymbol{x} \circ \boldsymbol{y}$.

A binary sequence $\boldsymbol{x} \in \Sigma_2^n$ is called *balanced* if the number of zeros and ones is identical, i.e., if $w(\boldsymbol{x}) = \frac{n}{2}$. We similarly define an *almost-balanced* binary sequence as follows.

Definition 1. A sequence $\boldsymbol{x} \in \Sigma_2^n$ is called $\varepsilon(n)$-*almost balanced* if $w(\boldsymbol{x}) \in \left[\frac{n}{2} - \varepsilon(n), \frac{n}{2} + \varepsilon(n)\right]$.

To extend the definition of balanced and almost balanced sequences to non-binary sequences we need the following additional notation. For $\sigma \in \Sigma_q$, let $\#_\sigma(\boldsymbol{x})$ denote the number of occurrences of the symbol σ in $\boldsymbol{x}$. A sequence $\boldsymbol{x} \in \Sigma_q^n$ is called *symbol-balanced* if any symbol $\sigma \in \Sigma_q$ appears in $\boldsymbol{x}$

exactly $\frac{n}{q}$ times. That is, $\#_\sigma(\boldsymbol{x}) = \frac{n}{q}$ for any $\sigma \in \Sigma_q$. When q is even, we say that $\boldsymbol{x}$ is *polarity-balanced* if

$$\sum_{i=0}^{\frac{q}{2}-1} \#_i(\boldsymbol{x}) = \sum_{i=\frac{q}{2}}^{q-1} \#_i(\boldsymbol{x}) = \frac{n}{2}.$$

Definition 1 can be extended to *α-almost symbol-balanced* and *α-almost polarity-balanced* as follows.

Definition 2. A sequence $\boldsymbol{x}$ is called *$\varepsilon(n)$-almost symbol-balanced* if for any $\sigma \in \Sigma_q$ we have that

$$\#_\sigma(\boldsymbol{x}) \in \left[\frac{n}{q} - \varepsilon(n), \frac{n}{q} + \varepsilon(n)\right].$$

Definition 3. A sequence $\boldsymbol{x}$ is called *$\varepsilon(n)$-almost polarity-balanced* if

$$\left| \sum_{i=0}^{\frac{q}{2}-1} \#_i(\boldsymbol{x}) - \sum_{i=\frac{q}{2}}^{q-1} \#_i(\boldsymbol{x}) \right| \leq 2 \cdot \varepsilon(n).$$

B. Related Work

In this work we extend our previous work [18] focusing on eliminating windows with small periods. This work utilizes a graph-based reduction technique to establish efficiency and convergence of the construction. Inspired by this, the current study proposes an iterative method for encoding sequences into almost-balanced ones without requiring monotonic progress between the algorithm's steps. In a parallel effort [19], [20], the technique is extended to address diverse constraints. A universal approach is presented and a general methodology for combining constraints is detailed, showcasing the versatility and comprehensive nature of the encoding framework.

C. Arithmetic Coding

Arithmetic coding [21] serves as a data compression method wherein the encoding process transforms an input sequence into a new sequence, representing a fractional value in the interval $[0, 1)$. Each iteration processes a single symbol from the input, dividing the current interval and designating one of the resulting partitions as the new interval. Consequently, the algorithm progressively operates on smaller intervals, and the output exists within each of these nested intervals.

One of the main components in our suggested construction is an encoder and decoder pair which are based on binary arithmetic coding. For the completeness of the results in the paper, the key concepts of binary arithmetic coding, which will be used throughout this paper, are described next.[1]

Let $p \in (0, 1)$ and let n be an integer. The encoding of a sequence $\boldsymbol{x}$ of length n is done by mapping $\boldsymbol{x}$ into a unique interval $I_{\boldsymbol{x}} \subseteq [0, 1)$ as follows.

1) Initialize $I \leftarrow [0, 1)$.
2) For $i = 1, 2, \ldots, n$:
   2.1. Split the interval I into two sub-intervals, I_L and I_R, of sizes $|I_L| = p \cdot |I|$ and $|I_R| = (1 - p) \cdot |I|$.
   2.2. If $x_i = 0$, $I \leftarrow I_L$
   2.3. Else, $I \leftarrow I_R$
3) $I_{\boldsymbol{x}} = I$.

[1]This description highlights the details necessary for our derivations and it can be considered a simplification of the standard arithmetic coding technique.

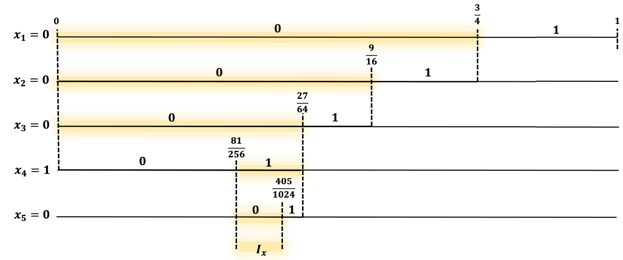

Figure 1. Mapping of $\boldsymbol{x} = 00010$ into an interval $I_{\boldsymbol{x}}$ for $p = \frac{3}{4}$ (and $n = 5$).

Finally, the encoding of $\boldsymbol{x}$ is the binary representation[2] of the shortest (fewest number of bits) fraction in the interval $I_{\boldsymbol{x}}$.

After mapping the input to an interval, the output of the algorithm is the binary representation of a fraction within the interval with a minimal number of bits in its binary representation. Given $p \in (0, 1)$ and an integer $n > 0$ we denote the corresponding encoder and decoder pair by $f_p^{(\mathbf{ac})} : \Sigma_2^n \to \Sigma_2^*$ and $g_p^{(\mathbf{ac})} : \Sigma_2^* \to \Sigma_2^n$.

Example 1. Figure 1 presents the described steps for the mapping of $\boldsymbol{x} = 00010$ into an interval $I_{\boldsymbol{x}}$ for $p = \frac{3}{4}$ (and $n = 5$). In this example, $I_{\boldsymbol{x}} = \left[\frac{81}{256}, \frac{405}{1024}\right)$, and the fraction with the minimal representation within this interval is $0.375 = 2^{-2} + 2^{-3}$, resulting with the output 011.

We note that the mapping of a sequence $\boldsymbol{x} \in \Sigma_2^n$ into an interval $I_{\boldsymbol{x}}$ requires splitting the interval n times, where each iteration computes the new interval edges in $\mathcal{O}(1)$ time. Hence, the worse-case time complexity of $f_p^{(\mathbf{ac})}$ and $g_p^{(\mathbf{ac})}$ is $\Theta(n)$.

III. Binary Almost Balanced Sequences

In this section, we discuss the case of binary $\varepsilon(n)$-almost balanced sequences. We first explicitly define the constraint as follows,

$$\mathcal{C}(n, \varepsilon(n)) = \left\{ \boldsymbol{x} \in \Sigma_2^n \;\middle|\; w(\boldsymbol{x}) \in \left[\frac{n}{2} - \varepsilon(n), \frac{n}{2} + \varepsilon(n)\right] \right\}.$$

In the next lemma, we show that there exists a single-redundancy-bit construction for $\mathcal{C}(n, \varepsilon(n))$ only if $\varepsilon(n) = \Omega(\sqrt{n})$. More precisely, for $\varepsilon(n) = \alpha\sqrt{n}$, we give lower and upper bounds on the minimal value of α for which there exists a single-redundancy-bit construction for $\mathcal{C}(n, \alpha\sqrt{n})$ for n large enough. More formally, for every $\alpha > 0$ we define $F(n, \alpha) \triangleq \frac{|\mathcal{C}(n, \alpha\sqrt{n})|}{2^n}$. Thus, our goal is to find the minimum α for which there exists n' such that for any $n \geq n'$ we have that $F(n, \alpha) \geq 1/2$.

Lemma 1. There exists a constant c such that if $\alpha \geq c$ and n is large enough, then there exists a single redundancy bit construction for $\mathcal{C}(n, \alpha\sqrt{n})$. Otherwise, if $\alpha < c$ then there is no such a construction. Moreover, it holds that $0.335 < c \leq 0.34$.

Proof. This result can be seen from the fact that the binomial distribution approaches the normal distribution as $n \to \infty$, with $\mu = n/2$ and $\sigma = \sqrt{n}/2$. Considering the Z-score

[2]Here we consider the binary representation of a fraction in $[0, 1)$ as the binary vector representing the corresponding sum of negative powers of two (similar to the representation of a positive integer, but with negative powers). For example 0.25 is represented by 01 and 0.75 is represented by 11.

table [22] we know that at least half of the space is thus contained in

$$[\mu - 0.68\sigma, \mu + 0.68\sigma] = [n/2 - 0.34\sqrt{n}, n/2 + 0.34\sqrt{n}].$$

On the other hand, the interval

$$[\mu - 0.67\sigma, \mu + 0.67\sigma] = [n/2 - 0.335\sqrt{n}, n/2 + 0.335\sqrt{n}]$$

contains strictly less than half of the space. □

Nonetheless, the problem of finding such a construction has remained unsolved as the state-of-the-art construction [17] with a single redundancy bit only tackles a linear-almost-balanced version of $[np_1, np_2]$ for $p_1 < 1/2 < p_2$. It should be noted that the construction in [17] utilizes the existence of mappings without explicitly determining them. Next, we demonstrate an efficient construction with a single redundancy bit for $\alpha > \sqrt{\ln(2)} \approx 0.8325$, inspired by the approach taken in [18] and based on the arithmetic coding [21] technique. To this end, we also define the two following auxiliary constraints,

$$\mathcal{C}_L(n, \alpha\sqrt{n}) = \left\{ \boldsymbol{x} \in \Sigma_2^n \;\middle|\; w(\boldsymbol{x}) \leq \frac{n}{2} + \alpha\sqrt{n} \right\},$$

$$\mathcal{C}_H(n, \alpha\sqrt{n}) = \left\{ \boldsymbol{x} \in \Sigma_2^n \;\middle|\; w(\boldsymbol{x}) \geq \frac{n}{2} - \alpha\sqrt{n} \right\}.$$

Notice that $\mathcal{C}(n, \alpha\sqrt{n}) = \mathcal{C}_L(n, \alpha\sqrt{n}) \cap \mathcal{C}_H(n, \alpha\sqrt{n})$. We now propose the overall construction as follows,

Construction 1 (Binary almost-balanced)**.** Let $\alpha > \sqrt{\ln(2)}$, let n be a sufficiently large[3] integer, and let $\boldsymbol{x} \in \{0,1\}^{n-1}$. Our construction is composed of the following two instances of the arithmetic coding described in Section II-C:

- Binary arithmetic coding with $p_L = 1/2 + \alpha/\sqrt{n} + 1/n$ and a pair of encoder and decoder functions $f_{p_L}^{(\mathbf{ac})} : \Sigma_2^n \to \Sigma_2^*$, $g_{p_L}^{(\mathbf{ac})} : \Sigma_2^* \to \Sigma_2^n$.
- Binary arithmetic coding with $p_H = 1/2 - \alpha/\sqrt{n} - 1/n$ and a pair of encoder and decoder functions $f_{p_H}^{(\mathbf{ac})} : \Sigma_2^n \to \Sigma_2^*$, $g_{p_H}^{(\mathbf{ac})} : \Sigma_2^* \to \Sigma_2^n$.

For simplicity, we assume that the output length of $f_{p_L}^{(\mathbf{ac})}, f_{p_H}^{(\mathbf{ac})}$ is at least $n-2$ (otherwise we can pad the output with zeros and we will show that it will be exactly $n-2$). Then, Algorithms 1 and 2 construct an efficient construction with a single redundancy bit and $\mathcal{O}(T(n))$ average time complexity for $T(n)$ the maximum time complexity amongst $f_{p_L}^{(\mathbf{ac})}, g_{p_L}^{(\mathbf{ac})}, f_{p_H}^{(\mathbf{ac})}, g_{p_H}^{(\mathbf{ac})}$. That is, the average time complexity is $\mathcal{O}(n)$.

The correctness of this construction is stated in the next theorem.

Theorem 1. Construction 1 is an efficient construction with a single redundancy bit for $\mathcal{C}(n, \alpha\sqrt{n})$ when $\alpha > \sqrt{\ln(2)}$ and n is large enough.

The proof of the theorem follows immediately from the following three lemmas.

Lemma 2. Algorithm 1 stops with an output $\boldsymbol{y} \in \mathcal{C}(n, \alpha\sqrt{n})$.

Proof. First, let us prove that in each iteration of Algorithm 1, the length of $\boldsymbol{y}$, is exactly n. Clearly, before entering the while

[3]While the size of n depends on α, the size of n for which the construction work is not too large. For example, for $\alpha = 1$ and $\alpha = 0.835$ we only need $n > 4$ and $n > 6$, respectively

Algorithm 1 Almost-Balanced Encoder E

Input: $\boldsymbol{x} \in \Sigma_2^{n-1}$.
Output: $\boldsymbol{y} \in \mathcal{C}(n, \alpha\sqrt{n})$.
1: $\boldsymbol{y} \leftarrow \boldsymbol{x} \circ 0$.
2: **while** $\boldsymbol{y} \notin \mathcal{C}(n, \alpha\sqrt{n})$ **do**
3: **if** $\boldsymbol{y} \in \mathcal{C}_L(n, \alpha\sqrt{n})$ **then**
4: $\boldsymbol{y} \leftarrow f_{p_L}^{(\mathbf{ac})}(\boldsymbol{y}) \circ 11$.
5: **else**
6: $\boldsymbol{y} \leftarrow f_{p_H}^{(\mathbf{ac})}(\boldsymbol{y}) \circ 01$.
7: **return** $\boldsymbol{y}$.

Algorithm 2 Almost-Balanced Decoder D

Input: $\boldsymbol{y} \in \mathcal{C}(n, \alpha\sqrt{n})$ such that $E(\boldsymbol{x}) = \boldsymbol{y}$ for $\boldsymbol{x} \in \Sigma_2^{n-1}$.
Output: $\boldsymbol{x} \in \Sigma_2^{n-1}$.
1: **while** $y_n \neq 0$ **do**
2: **if** $y_{n-1} = 1$ **then**
3: $\boldsymbol{y} \leftarrow g_{p_L}^{(\mathbf{ac})}(\boldsymbol{y}_{[1:n-2]})$.
4: **if** $y_{n-1} = 0$ **then**
5: $\boldsymbol{y} \leftarrow g_{p_H}^{(\mathbf{ac})}(\boldsymbol{y}_{[1:n-2]})$.
6: **return** $\boldsymbol{y}_{[1:n-1]}$.

loop for the first time, the length of $\boldsymbol{y}$ is n. Within the while loop, $\boldsymbol{y}$ is modified to be the concatenation of either $f_{p_L}^{(\mathbf{ac})}(\boldsymbol{y})$ or $f_{p_H}^{(\mathbf{ac})}(\boldsymbol{y})$ with two additional bits. Hence, we need to show that the output of $f_{p_L}^{(\mathbf{ac})}(\boldsymbol{y})$ or $f_{p_H}^{(\mathbf{ac})}(\boldsymbol{y})$, respectively, is of length $n-2$. Recall that by our assumption, the output length of $f_{p_L}^{(\mathbf{ac})}, f_{p_H}^{(\mathbf{ac})}$ is always at least $n-2$. Hence, it is sufficient to show that the length cannot be greater than $n-2$.

If $\boldsymbol{y}$ is modified in Step 4 then $\boldsymbol{y} \notin \mathcal{C}(n, \alpha\sqrt{n})$ and $\boldsymbol{y} \in \mathcal{C}_L(n, \alpha\sqrt{n})$. That is, $w(\boldsymbol{y}) < \frac{n}{2} - \alpha\sqrt{n}$, and the length of the interval that corresponds to $\boldsymbol{y}$ by the mapping $f_{p_L}^{(\mathbf{ac})}$ is

$$|I_{\boldsymbol{y}}| = \left(\frac{1}{2} - \frac{\alpha}{\sqrt{n}} - \frac{1}{n}\right)^{w(\boldsymbol{y})} \cdot \left(\frac{1}{2} + \frac{\alpha}{\sqrt{n}} + \frac{1}{n}\right)^{n-w(\boldsymbol{y})}.$$

Since this value is minimized when $w(\boldsymbol{y})$ is maximized, we have that any such $\boldsymbol{y}$ is mapped to an interval of length

$$|I_{\boldsymbol{y}}| \geq \left(\frac{1}{2} - \frac{\alpha}{\sqrt{n}} - \frac{1}{n}\right)^{\frac{n}{2} - \alpha\sqrt{n}} \cdot \left(\frac{1}{2} + \frac{\alpha}{\sqrt{n}} + \frac{1}{n}\right)^{\frac{n}{2} + \alpha\sqrt{n}}.$$

Moreover, since

$$\lim_{n\to\infty} 2^{n-2} \cdot \left(\frac{1}{2} - \frac{\alpha}{\sqrt{n}} - \frac{1}{n}\right)^{\frac{n}{2} - \alpha\sqrt{n}} \cdot \left(\frac{1}{2} + \frac{\alpha}{\sqrt{n}} + \frac{1}{n}\right)^{\frac{n}{2} + \alpha\sqrt{n}} = \frac{e^{2\alpha^2}}{4},$$

it can be verified that for any $\alpha > \sqrt{\ln(2)}$ we have that $|I_{\boldsymbol{y}}| \geq 1/2^{n-2}$ for n large enough. This implies that it is possible to enumerate the interval with exactly $n-2$ symbols.

By the definition of Algorithm 1, if the algorithm ends, it stops with a valid output. Hence, it is left to show that the algorithm converges. Similarly to the approach presented in [18], the convergence follows from a reduction to an acyclic graph walk, and it is given here for completeness.

Let $G = (V, \mathcal{E})$ be a directed graph such that $V = \Sigma_2^n$ is the set of nodes and the set of edges $\mathcal{E} \subseteq V \times V$ is defined as follows. From any $\boldsymbol{u} \notin \mathcal{C}(n, \alpha\sqrt{n})$ there is a single outgoing edge to the node $\boldsymbol{v} \in V$, where $\boldsymbol{v} = f_{p_L}^{(\mathbf{ac})}(\boldsymbol{u}) \circ 11$ if $\boldsymbol{y} \in \mathcal{C}_L(n, \alpha\sqrt{n})$, and otherwise $\boldsymbol{v} = f_{p_H}^{(\mathbf{ac})}(\boldsymbol{u}) \circ 01$. That is, there is an edge from node $\boldsymbol{u}$ to node $\boldsymbol{v}$ if $\boldsymbol{v}$ is the unique sequence such that the operations inside the while loop of Algorithm 1 will modify $\boldsymbol{y}=\boldsymbol{u}$ to $\boldsymbol{y}=\boldsymbol{v}$. Note that the mappings $f_{p_L}^{(\mathbf{ac})}$ and $f_{p_H}^{(\mathbf{ac})}$ are invertible functions and hence the in-degree of all the nodes in V is at most one. Moreover, by the definition of $\mathcal{E}$, any node $\boldsymbol{v}\in V$ for which $v_n=1$, satisfies $d_{in}(\boldsymbol{v})=0$.

Assume by contradiction that the encoder does not converge for an input $\boldsymbol{x}\in\Sigma_2^{n-1}$ and let

$$\boldsymbol{x}\circ 0=\boldsymbol{y}^{(0)},\boldsymbol{y}^{(1)},\boldsymbol{y}^{(2)},\dots$$

be the list of nodes that correspond to the values of $\boldsymbol{y}$ before each iteration of the while loop in Algorithm 1. Since $|V|$ is finite, the path $\boldsymbol{y}^{(0)}\to\boldsymbol{y}^{(1)}\to\boldsymbol{y}^{(2)}\to\cdots$ contains a cycle, i.e., there is an index i such that

$$\boldsymbol{y}^{(i)}\to\boldsymbol{y}^{(i+1)}\to\cdots\to\boldsymbol{y}^{(j-1)}\to\boldsymbol{y}^{(j)}=\boldsymbol{y}^{(i)}.$$

For a node $\boldsymbol{v}\in V$, let $d_{in}(\boldsymbol{v})$ be the in-degree of $\boldsymbol{v}$. Since $y_n^{(0)}=0$, we have that $d_{in}(\boldsymbol{y}^0)=0$ and hence $\boldsymbol{y}^0$ is not on the cycle. Hence, there exists an index i' such that $d_{in}(\boldsymbol{y}^{(i')})=2$, which is a contradiction. □

Example 2. Let $n=8,\alpha=1$ and $\boldsymbol{x}=1000000\in\Sigma_2^7$. Note that $\mathcal{C}(8,\sqrt{8})=\{\boldsymbol{y}\in\Sigma_2^8 \mid 2\le w(\boldsymbol{y})\le 6\}$. Using the same notations as in the proof of Lemma 2, Algorithm 1 begins with $\boldsymbol{y}^{(0)}=\boldsymbol{x}\circ 0=1000000\circ 0$. As $\boldsymbol{y}^{(0)}\notin\mathcal{C}(8,\sqrt{8})$ the algorithm enters the while loop and line 4 is executed ($w(\boldsymbol{y}^{(0)})\le 6$). It can be verified that in this step $\boldsymbol{y}^{(0)}$ is mapped to the interval $[0.97855,0.99698)$ which can be represented using 111111. Hence $\boldsymbol{y}^{(1)}=111111\circ 11$. In the second iteration, line 6 is executed and $\boldsymbol{y}^{(2)}=100000\circ 01\in\mathcal{C}(8,\sqrt{8})$.

Lemma 3. For any $\boldsymbol{x}\in\Sigma_2^{n-1}$, the decoder D from Algorithm 2 satisfies $D\left(E(\boldsymbol{x})\right)=\boldsymbol{x}$.

Lemma 4. The average number of iterations in the while loop of E,D is at most $|\Sigma|=O(1)$.

The proofs of Lemma 3 and Lemma 4 follow from the reduction of the encoder to a graph walk presented in the proof of Lemma 2. Since the proofs are very similar to the ones presented in [19], they are omitted.

IV. Extensions to Non-binary Alphabets

In this section we discuss the extension of Construction 1 to the non-binary case.

A. Almost Polarity-Balanced

Construction 1 can be modified to obtain almost polarity-balanced sequences for any alphabet of even size. First, we formally define the constraint

$$\mathcal{C}_q^{(\mathbf{pb})}(n,\varepsilon(n))=\left\{\boldsymbol{x}\in\Sigma_q^n \;\middle|\; \sum_{i=0}^{\frac{q}{2}-1}\#_i(\boldsymbol{x})\in\left[\frac{n}{2}-\varepsilon(n),\frac{n}{2}+\varepsilon(n)\right]\right\}.$$

Similarly to the binary case, we are interested in single redundancy symbol codes and in the next lemma we show that $\varepsilon(n)=\Omega(\sqrt{n})$. To this end, we define $F^{(\mathbf{pb})}(n,\alpha)\triangleq|\mathcal{C}_q^{(\mathbf{pb})}(n,\alpha\sqrt{n})|/q^n$.

Lemma 5. Let $c=\liminf\{\alpha|F^{(\mathbf{pb})}(n,\alpha)\ge 1/q\}$. Then, it holds that $c\le 0.34$.

Proof: It holds that

$$F^{(\mathbf{pb})}(n,\alpha)=\frac{|\mathcal{C}_q^{(\mathbf{pb})}(n,\alpha\sqrt{n})|}{q^n}=\frac{|\mathcal{C}(n,\alpha\sqrt{n})|\cdot(q/2)^n}{q^n}=\frac{|\mathcal{C}(n,\alpha\sqrt{n})|}{2^n}=F(n,\alpha).$$

The latter together with Lemma 1 completes the proof. ■

While the bound in the previous lemma is not tight, knowing that $F^{(\mathbf{pb})}(n,\alpha)=F(n,\alpha)$, we can derive tighter upper and lower bounds using the same techniques as in the proof of Lemma 1. Table I summarizes the upper and lower bounds on c for small alphabet size, that can be derived by this manner.

Table I. Bounds on $c=\liminf\{\alpha|F^{(\mathbf{pb})}(n,\alpha)\ge 1/q\}$.

q	Lower bound	Upper bound
2	0.335	0.34
3	0.215	0.22
4	0.155	0.16
5	0.125	0.13
6	0.105	0.11
7	0.09	0.095

Similarly to Construction 1, we define $\mathcal{C}_q^{(\mathbf{pb})}(n,\alpha\sqrt{n})$ as the intersection of the following two constraint channels,

$$\mathcal{C}_{q,L}^{(\mathbf{pb})}(n,\alpha\sqrt{n})=\left\{\boldsymbol{x}\in\Sigma_q^n \;\middle|\; \sum_{i=0}^{\frac{q}{2}-1}\#_i(\boldsymbol{x})\le\frac{n}{2}+\alpha\sqrt{n}\right\},$$

$$\mathcal{C}_{q,H}^{(\mathbf{pb})}(n,\alpha\sqrt{n})=\left\{\boldsymbol{x}\in\Sigma_q^n \;\middle|\; \sum_{i=0}^{\frac{q}{2}-1}\#_i(\boldsymbol{x})\ge\frac{n}{2}-\alpha\sqrt{n}\right\}.$$

To address non-binary alphabets, we modify our arithmetic coding based mappings as follows. First, in each iteration we partition the current interval into q sub-intervals (instead of two) and if the current symbol is i, we continue with the i-th sub-interval to the next iteration. For $p\in(0,1)$ we define the size of the first $\frac{q}{2}$ sub-intervals to be $\frac{2p}{q}|I|$, and the size of the last $\frac{q}{2}$ sub-intervals to be $\frac{2(1-p)}{q}|I|$, where I is the current interval. Given $p\in(0,1)$, an integer $n>0$, and an even integer $q\ge 2$, we denote the corresponding encoder and decoder pair by $f_{q,p}^{(\mathbf{pb\text{-}ac})}:\Sigma_q^n\to\Sigma_q^*$ and $g_{q,p}^{(\mathbf{pb\text{-}ac})}:\Sigma_q^*\to\Sigma_q^n$.

Construction 2 (Almost polarity balanced)**.** For an even integer $q\ge 2$, let $\alpha>\sqrt{\ln(q)}$, let n be a sufficiently large integer, and let $\boldsymbol{x}\in\Sigma_q^{n-1}$. Our construction is composed of the following two instances of the modified arithmetic coding:

- q-ary arithmetic coding with $p_L=1/2+\alpha/\sqrt{n}+1/n$ and a pair of encoder and decoder functions $f_{q,p_L}^{(\mathbf{pb\text{-}ac})}:\Sigma_q^n\to\Sigma_q^*$, $g_{q,p_L}^{(\mathbf{pb\text{-}ac})}:\Sigma_q^*\to\Sigma_q^n$.
- q-ary arithmetic coding with $p_H=1/2-\alpha/\sqrt{n}-1/n$ and a pair of encoder and decoder functions $f_{q,p_H}^{(\mathbf{pb\text{-}ac})}:\Sigma_q^n\to\Sigma_q^*$, $g_{q,p_H}^{(\mathbf{pb\text{-}ac})}:\Sigma_q^*\to\Sigma_q^n$.

Then Algorithms 1 and 2 (with the corresponding modifications) construct an efficient construction with a single redundancy symbol and $\mathcal{O}(T(n))$ average time complexity, where $T(n)$ is the maximum complexity amongst $f_{q,p_L}^{(\mathbf{pb\text{-}ac})},g_{q,p_L}^{(\mathbf{pb\text{-}ac})},f_{q,p_H}^{(\mathbf{pb\text{-}ac})},g_{q,p_H}^{(\mathbf{pb\text{-}ac})}$.

The correctness of Construction 2 follows from the same arguments that were presented in the proof of Construction 1.

Hence, it is sufficient to show that the length of $\boldsymbol{y}$ before each iteration of the while-loop is n. That is, we need to show that

$$\left(\frac{1}{q}-\frac{2\alpha}{q\sqrt{n}}-\frac{2}{qn}\right)^{\frac{n}{2}-\alpha\sqrt{n}}\cdot\left(\frac{1}{q}+\frac{2\alpha}{q\sqrt{n}}+\frac{2}{qn}\right)^{\frac{n}{2}+\alpha\sqrt{n}} \geq \frac{1}{q^{n-2}},$$

which holds for any $\alpha > \sqrt{\ln(q)}$. We note that for $q = 2$ Construction 2 is identical to Construction 1. However, for $q > 2$ we can improve Construction 2 by noticing that now instead of using two bits it is enough to use one symbol in order to indicate the three options of determining when the decoder stops and whether to decode with $g^{(\textbf{pb-ac})}_{q,p_L}$ or $g^{(\textbf{pb-ac})}_{q,p_H}$. Hence, we can change Algorithm 1 such that in step 4 we assign $f^{(\textbf{pb-ac})}_{q,p_L}(\boldsymbol{y}) \circ 1$ into $\boldsymbol{y}$ and in step 6 we assign $f^{(\textbf{pb-ac})}_{q,p_L}(\boldsymbol{y}) \circ 2$ into $\boldsymbol{y}$. Accordingly, we modify step 2 and step 5 of Algorithm 2 to check whether y_n equals 1 or 2, respectively. By doing so, we allow the output of $f^{(\textbf{pb-ac})}_{q,p_L}$ and $f^{(\textbf{pb-ac})}_{q,p_H}$ to be of length $n-1$. Hence, we need

$$\left(\frac{1}{q}-\frac{2\alpha}{q\sqrt{n}}-\frac{2}{qn}\right)^{\frac{n}{2}-\alpha\sqrt{n}}\cdot\left(\frac{1}{q}+\frac{2\alpha}{q\sqrt{n}}+\frac{2}{qn}\right)^{\frac{n}{2}+\alpha\sqrt{n}} \geq \frac{1}{q^{n-1}},$$

which holds for $\alpha > \sqrt{\frac{\ln(q)}{2}}$.

A more thorough examination, reveals that the latter construction can be further improved for larger values of q. For such values of q, we do not need $f^{(\textbf{pb-ac})}_{q,p_L}$ and $f^{(\textbf{pb-ac})}_{q,p_H}$ to compress the input at all. For our purposes, it is sufficient to only restrict the values of the last symbol in the output and use the remaining symbols to distinguish between the states.

B. Almost symbol-balanced

Lastly, we discuss $\varepsilon(n)$-almost symbol-balanced q-ary sequences. For simplicity, we only give a construction for $q = 4$ while the generalization to $q = 2^\ell$ for any positive integer ℓ is straightforward. Given n and ε we define the constraint as,

$$\mathcal{C}^{(\textbf{sb})}_4(n,\varepsilon(n))=\left\{\boldsymbol{x} \in \Sigma_4^n \;\middle|\; \#_\sigma(\boldsymbol{x})\in\left[\frac{n}{4}-\varepsilon(n), \frac{n}{4}+\varepsilon(n)\right], \forall\sigma \in \Sigma_4\right\}.$$

We start by noting that any $\varepsilon(n)$-almost symbol-balanced sequence is also a $\varepsilon(n)$-almost polarity-balanced. Hence, our analysis of almost polarity-balanced codes implies that if a single-redundancy-symbol $\varepsilon(n)$-almost symbol-balanced code exists then $\varepsilon(n) = \Omega(\sqrt{n})$. Therefore, we focus again on the case where $\varepsilon(n) = \alpha\sqrt{n}$.

Our construction is based on defining a subset of $\mathcal{C}^{(\textbf{sb})}_4(n, \alpha\sqrt{n}))$ as the intersection of the following three $\frac{\alpha\sqrt{n}}{2}$-polarity-balanced 4-ary codes,

$$\mathcal{C}^{(\textbf{pb})}_{0,i}=\left\{\boldsymbol{x} \in \Sigma_4^n \;\middle|\; \#_0(\boldsymbol{x}) + \#_i(\boldsymbol{x})\in\left[\frac{n}{2}-\frac{\alpha\sqrt{n}}{2}, \frac{n}{2}+\frac{\alpha\sqrt{n}}{2}\right]\right\},$$

for $i \in \{1, 2, 3\}$.

Lemma 6. It holds that $\mathcal{C}^{(\textbf{pb})}_{0,1} \cap \mathcal{C}^{(\textbf{pb})}_{0,2} \cap \mathcal{C}^{(\textbf{pb})}_{0,3} \subseteq \mathcal{C}^{(\textbf{sb})}_4(n, \alpha\sqrt{n})$.

Construction 3 (Almost symbol balanced)**.** Let $\alpha > 2\sqrt{\ln(4)}$, let n be a sufficiently large integer, and let $\boldsymbol{x} \in \Sigma_q^{n-1}$. Our construction is composed of three pairs of the modified arithmetic coding that were utilized in Construction 2. For each $i \in \{1, 2, 3\}$ consider $\mathcal{C}^{(\textbf{pb})}_{0,i}$ and define:

- 4-ary arithmetic coding with $p_L = 1/2 + \alpha/2\sqrt{n} + 1/n$ that associates the first two intervals in each partition with 0 and i, and a pair of encoder and decoder functions $f^{(\textbf{pb-ac})}_{4,p_L,i} : \Sigma_4^n \to \Sigma_4^*$, $g^{(\textbf{pb-ac})}_{4,p_L,i} : \Sigma_4^* \to \Sigma_4^n$.
- 4-ary arithmetic coding with $p_H = 1/2 - \alpha/2\sqrt{n} - 1/n$ that associates the first two intervals in each partition with 0 and i, and a pair of encoder and decoder functions $f^{(\textbf{pb-ac})}_{4,p_H,i} : \Sigma_4^n \to \Sigma_4^*$, $g^{(\textbf{pb-ac})}_{4,p_H,i} : \Sigma_4^* \to \Sigma_4^n$.
- The constraint channels

$$\mathcal{C}^{(\textbf{pb})}_{L,0,i} = \left\{\boldsymbol{x} \in \Sigma_4^n \;\middle|\; \#_0(\boldsymbol{x}) + \#_i(\boldsymbol{x}) \leq \frac{n}{2} + \frac{\alpha\sqrt{n}}{2}\right\}$$

$$\mathcal{C}^{(\textbf{pb})}_{H,0,i} = \left\{\boldsymbol{x} \in \Sigma_4^n \;\middle|\; \#_0(\boldsymbol{x}) + \#_i(\boldsymbol{x}) \geq \frac{n}{2} - \frac{\alpha\sqrt{n}}{2}\right\}$$

Then Algorithms 3 and 4 construct an efficient construction with a single redundancy symbol and $\mathcal{O}(T(n))$ average time complexity, where $T(n)$ is the maximum complexity amongst $f^{(\textbf{pb-ac})}_{4,p_L,i}, f^{(\textbf{pb-ac})}_{4,p_H,i}, g^{(\textbf{pb-ac})}_{4,p_L,i}, g^{(\textbf{pb-ac})}_{4,p_H,i}$ for $i \in \{1, 2, 3\}$.

Algorithm 3 Almost-Balanced 4-ary Encoder E_4

Input: $\boldsymbol{x} \in \Sigma_4^{n-1}$.
Output: $\boldsymbol{y} \in \mathcal{C}^{(\textbf{sb})}_4(n, \alpha\sqrt{n})$.

```
y ← x ∘ 0.
while y ∉ C_4^(sb)(n, α√n) do
    for i ∈ {1, 2, 3} do
        if y ∉ C_{0,i}^(pb) then
            if y ∉ C_{L,0,i}^(pb) then
                y ← f_{4,p_L,i}^(pb-ac)(y) ∘ 1 ∘ i.
            else
                y ← f_{4,p_H,i}^(pb-ac)(y) ∘ 0 ∘ i.
        break
return y.
```

Algorithm 4 Almost-Balanced 4-ary Decoder D_4

Input: $\boldsymbol{y} \in \mathcal{C}^{(\textbf{sb})}_4(n, \alpha\sqrt{n})$ such that $E_4(\boldsymbol{x}) = \boldsymbol{y}$ for some $\boldsymbol{x} \in \Sigma_4^{n-1}$.
Output: $\boldsymbol{x} \in \Sigma_4^{n-1}$.

```
while y_n ≠ 0 do
    if y_{n-1} = 1 then
        y ← g_{4,p_L,y_n}^(pb-ac)(y_[1:n-2]).
    if y_{n-1} = 0 then
        y ← g_{4,p_H,y_n}^(pb-ac)(y_[1:n-2]).
return y_[1:n-1].
```

The correctness of Construction 3 follows from arguments similar to the those presented in the proof of Construction 1, and the observation that for $i \in \{1, 2, 3\}$ the output of $f^{(\textbf{pb-ac})}_{4,p_H,i}$ and $f^{(\textbf{pb-ac})}_{4,p_H,i}$ is of length $n-2$ for $\alpha > 2\sqrt{\ln(4)}$.

V. Conclusion

This work studies the problem of encoding almost-balanced sequences using a single redundancy symbol. While our constructions achieve an optimally balanced order of $\Theta(\sqrt{n})$, there persists a multiplicative gap between theoretical bounds on $\varepsilon(n)$ and the values applicable in our constructions. Furthermore, bounding the worst case time complexity of the algorithms is a challenging problem which is left for future work. Experimental results verified that the number of iterations of Algorithm 1 is at most 7 for words of length $n = 30$.

REFERENCES

[1] D. Knuth, "Efficient balanced codes," *IEEE Transactions on Information Theory*, vol. 32, no. 1, pp. 51–53, 1986.

[2] M. Blawat, K. Gaedke, I. Huetter, X.-M. Chen, B. Turczyk, S. Inverso, B. W. Pruitt, and G. M. Church, "Forward error correction for DNA data storage," *Procedia Computer Science*, vol. 80, pp. 1011–1022, 2016.

[3] R. N. Grass, R. Heckel, M. Puddu, D. Paunescu, and W. J. Stark, "Robust chemical preservation of digital information on DNA in silica with error-correcting codes," *Angewandte Chemie Int. Edition*, no. 8, pp. 2552–2555, Feb. 2015.

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

[5] S. M. H. T. Yazdi, R. Gabrys, and O. Milenkovic, "Portable and error-free DNA-based data storage," *Scientific reports*, vol. 7, no. 1, p. 5011, 2017.

[6] S. M. H. T. Yazdi, H. M. Kiah, E. Garcia-Ruiz, J. Ma, H. Zhao, and O. Milenkovic, "DNA-based storage: Trends and methods," *IEEE Transactions on Molecular, Biological and Multi-Scale Communications*, vol. 1, no. 3, pp. 230–248, 2015.

[7] L. Tallini and B. Bose, "Balanced codes with parallel encoding and decoding," *IEEE Transactions on Computers*, vol. 48, no. 8, pp. 794–814, 1999.

[8] L. Tallini, R. Capocelli, and B. Bose, "Design of some new efficient balanced codes," *IEEE Transactions on Information Theory*, vol. 42, no. 3, pp. 790–802, 1996.

[9] J. H. Weber and K. A. S. Immink, "Knuth's balanced codes revisited," *IEEE Transactions on Information Theory*, vol. 56, no. 4, pp. 1673–1679, 2010.

[10] K. A. Schouhamer Immink and J. H. Weber, "Very efficient balanced codes," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 2, pp. 188–192, 2010.

[11] L. G. Tallini and U. Vaccaro, "Efficient m-ary balanced codes," *Discrete Applied Mathematics*, vol. 92, no. 1, pp. 17–56, 1999.

[12] R. Mascella and L. Tallini, "On symbol permutation invariant balanced codes," in *Proceedings of the International Symposium on Information Theory (ISIT)*, 2005, pp. 2100–2104.

[13] ——, "Efficient m-ary balanced codes which are invariant under symbol permutation," *IEEE Transactions on Computers*, vol. 55, no. 8, pp. 929–946, 2006.

[14] T. G. Swart and J. H. Weber, "Efficient balancing of q-ary sequences with parallel decoding," in *IEEE International Symposium on Information Theory (ISIT)*, 2009, pp. 1564–1568.

[15] J. H. Weber, K. A. S. Immink, P. H. Siegel, and T. G. Swart, "Polarity-balanced codes," in *Information Theory and Applications Workshop (ITA)*, 2013, pp. 1–5.

[16] K. A. S. Immink, J. H. Weber, and H. C. Ferreira, "Balanced runlength limited codes using knuth's algorithm," in *Proceedings of the IEEE International Symposium on Information Theory (ISIT)*, 2011, pp. 317–320.

[17] T. T. Nguyen, K. Cai, and K. A. S. Immink, "Binary subblock energy-constrained codes: Knuth's balancing and sequence replacement techniques," in *IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2020, pp. 37–41.

[18] A. Kobovich, O. Leitersdorf, D. Bar-Lev, and E. Yaakobi, "Codes for constrained periodicity," in *IEEE International Symposium on Information Theory and its Applications (ISITA)*, 2022.

[19] ——, "Universal framework for parametric constrained coding," in *Proceedings of the IEEE International Symposium on Information Theory (ISIT)*, 2024.

[20] D. Bar-Lev, A. Kobovich, O. Leitersdorf, and E. Yaakobi, "Universal framework for parametric constrained coding," *arXiv preprint arXiv:2304.01317*, 2023.

[21] J. Rissanen and G. G. Langdon, "Arithmetic coding," *IBM Journal of research and development*, vol. 23, no. 2, pp. 149–162, 1979.

[22] E. Kreyszig, H. Kreyszig, and E. Norminton, *Advanced Engineering Mathematics*, 10th ed. John Wiley & Sons, 2011.