Universal Framework for Parametric Constrained Coding

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

***Abstract*—Constrained coding is a fundamental field in coding theory that tackles efficient communication through constrained channels. While fixed constraints (e.g., a fixed set of substrings may not appear in transmitted messages) have a general optimal solution, there is increasing demand for supporting *parametric* constraints that are dependent on the message length and portray some property that the substrings must satisfy (e.g., no $\log(n)$ consecutive zeros). Several works have tackled such parametric constraints through *iterative* algorithms following the sequence-replacement approach, yet this approach requires complex constraint-specific properties to guarantee convergence through *monotonic progression*. In this paper, we propose a universal framework for tackling *any* parametric constraint problem with far fewer requirements, through a *simple* iterative algorithm. By reducing an execution of this iterative algorithm to an acyclic graph traversal, we prove a surprising result that guarantees convergence with efficient average time complexity *even without requiring any monotonic progression*. We demonstrate how to apply this algorithm to the run-length-limited, minimal Hamming weight, local almost-balanced Hamming weight constraints, as well as repeat-free and secondary-structure constraints. Overall, this framework enables state-of-the-art results with minimal effort.**

I. Introduction

Constrained coding stands as a fundamental discipline within information theory, central to a myriad of applications ranging from communication systems to data storage [1]. At its core, constrained coding addresses the task of encoding information under specific constraints imposed by various communication channels and storage media.

Over the years, extensive research has yielded a universal framework [2] for constructing codes that adhere to *fixed constraints* – constraints that are independent of the message length (e.g., the sequence 1010 may not appear in the encoded message). The universal method constructs a deterministic finite automaton that only accepts words that satisfy the constraints, and then applies the state-splitting method [3] to devise an encoder/decoder pair. This method produces encoders with capacity-approaching rates.

Conversely, there is growing interest in *parametric constraints* that require the encoded messages to satisfy criteria that is parametric on the message length n. Such parametric constraints commonly restrict all windows of the encoded message with length $\ell(n)$ to satisfy some criteria; equivalently, all $\ell(n)$-substrings must not belong to a set $W(\ell(n))$ of the substrings that do not satisfy the criteria. Previous works [4]–[11] have often utilized iterative methods that replace forbidden substrings with alternate representations appended to the end of the message (e.g., *sequence replacement* [4]). However, these works typically struggle with guaranteeing the encoder convergence due to inter-dependence between substrings: replacing one forbidden substring with an appended alternate representation may render another forbidden substring (either at the end with the appended representation, or in the original location of the substring due to the merging of the symbols before and after the substring). Therefore, they design intricate modifications that are tailored for each constraint to ensure *monotonic progression*, leveraging properties of the constraint and additional symbols to limit new forbidden substrings. Hence, existing methods are often tailored to specific applications, requiring substantial effort when confronted with new constraints.

We have recently proposed an encoder for the constrained periodicity task that, in contrast to the previous iterative algorithms, guarantees encoder and decoder convergence without requiring monotonic progression [12]. Rather, we provided a graph-theory proof that essentially demonstrates that an encoder with an invertible step function will always converge and with complexity $O(1)$ steps on average. In this work, we generalize the method from [12] to tackle *any* parametric constraint which holds some basic requirements, leading to state-of-the-art results with minimal effort (see Table I). We demonstrate the effectiveness of the proposed method on the run-length-limited (RLL), minimal Hamming weight, and balanced Hamming weight constraints – improving both results and simplicity. Finally, we generalize beyond the study of forbidden windows to parametric constraints that may be dependent on the message itself to tackle the repeat-free and DNA secondary structure constraints.

II. Problem Statement

Parametric constraints are described according to a general formulation that generalizes to messages of different lengths. For example, a parametric constraint may be that messages of length n may not contain $\log(n)$ consecutive zeros. We generalize this as follows,

Definition 1 (Parametric Constraint)**.** A parametric constraint $\mathcal{C}(n)$ applied to a channel of length n is a channel that can only accept messages $\boldsymbol{y} \in \Sigma^n$ such that $\boldsymbol{y} \in \mathcal{C}(n)$.

The encoder and decoder for a parametric constrained code can be defined as follows.

The first two authors contributed equally to this work.

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.

Table I. Our framework achievable range and redundancy for different constraints in compression to previous works. The ℓ-RLL is tackled by [11] and Construction 1. ℓ, p-MW is tackled by [11] and Construction 2. For ℓ, p_1, p_2-AB between the work of [7] and Construction 3. ℓ-RF tackled by [13] and Construction 4. Lastly, for ℓ-SSA we compare between [14] and Construction 5.

	Previous Work		Our Work	
Constraint	# Red	Achievable Range	# Red	Achievable Range
ℓ-RLL	1	$\ell \geq \log(n)+1$	1	$\ell \geq \log(n)+1$
ℓ, p-MW	$p(n)$	$\ell \geq \lceil\log(n)\rceil + (p(n)-1)\cdot\lceil\log(\ell+2)\rceil + 2$	1	$\ell \geq \lceil\log(n)\rceil + (p(n)-1)\cdot\lceil\log(\ell+1)\rceil + 1$
ℓ, p_1, p_2-AB	1	$n \geq 16, \ell \geq 7, \ell \geq \frac{1}{c^2}\ln n + 2$	1	$n \geq 4, \ell \geq \frac{1}{c^2}\ln n$
$\ell - RF$	2	$\ell \geq 2\log(n)+2$	1	$\ell \geq 2\log(n)+1$
$\ell - SSA$	1	$n \geq 16, \ell \geq 6\log(n)+4$	1	$\ell \geq 2\log(n)+1$

Definition 2 (Constrained Code Encoder and Decoder)**.** A parametric constraint encoder $f_k : \Sigma^k \to \mathcal{C}(n)$ encodes messages of length k to messages of length $n = k + r$ that satisfy the parametric constraint $\mathcal{C}(n)$. A parametric constraint decoder $g_k : \mathcal{C}(n) \to \Sigma^k$ satisfies $g_k(f_k(\boldsymbol{x})) = x$ for any $\boldsymbol{x} \in \Sigma^k$, where the redundancy r and the message length k can be functions of n.

Our method addresses the challenges faced by previous works by proposing a novel construction to general parametric constraints. This unified approach aims to overcome the limitations of existing methods, offering an efficient framework for constrained coding.

Problem 1 (Universal Framework for Parametric Constrained Coding)**.** Devising a general method for constructing an efficient encoder and decoder for any parametric constraint.

While our method can be applied to any parametric constraint, we focus here on the binary alphabet codes, $\Sigma = \{0, 1\}$, with a single redundancy bit, $r = 1$. Additional cases are studied in the extended version of the paper [15].

Moreover, as the predominant focus of preceding research has centered on a specific subset of constraints termed substring-avoiding constraints, we have chosen to showcase our methodology within this context. By doing so, we aim to facilitate a comparative analysis with other relevant works. In contrast, explicit encoder and decoder for a constraint that falls outside the scope of substring-avoiding constraints are presented in our concurrent work [16].

Definition 3 (SA)**.** The $W(n)$*-substring-avoiding (SA)* constraint is a parametric constraint $\mathcal{C}(n)$ that includes all words of length n that do not contain any word $\boldsymbol{w} \in W(n)$ as a substring. We assume for simplicity that all $\boldsymbol{w} \in W(n)$ are of length $\ell = \ell(n)$.[1]

Problem 2 (Universal Framework for SA)**.** Constructing explicit encoders and decoders with a single redundancy bit for common SA channels.

III. Universal Framework for Constrained Windows

This section presents a construction for Problem 2, introducing an encoder (Algorithm 1) and decoder (Algorithm 2) with a single redundancy bit for any SA channel which satisfies the basic requirements presented in Theorem 1. The encoder is based on an iterative approach that searches for forbidden substrings within the current message and replaces them with an alternate encoding of the substring and its index appended to the end of the message (of length such that the overall message length is constant). Typically versions of this approach were utilized in previous sequence-replacement works, yet they expanded upon this naive encoder with additional constraint-specific symbols that aimed to guarantee monotonic progression (e.g., that the alternate representation does not create new forbidden substrings when appended to the previous message). Conversely, the proof from [12] that we generalize here to tackle the more general case of any constrained-window set $W(n)$ demonstrates that monotonic progression is not necessary for convergence if the simpler requirements of Theorem 1 are met.

To apply the universal framework for a specific $W(n)$-SA constraint, one should be able to identify whether a window $\boldsymbol{w}$ is in $W(n)$, which we denote as an indicator $\mathbf{1}_{W(n)} : \Sigma^\ell \to \{0, 1\}$, and to be able to represent each constraint-violating window using $\ell' = \ell(n) - \lceil\log(n)\rceil - 1$ symbols, with an injective function $\mathcal{R} : W(n) \to \Sigma^{\ell'}$. Where ℓ and ℓ' are always functions of n in the context of parametric constraints. Using $\mathbf{1}_{W(n)}$ we can create $\mathbf{1}_{\mathcal{C}(n)} : \Sigma^n \to \{0, 1\}$ as a function that returns 1 if neither of the $n-\ell+1$ windows in the message belongs to $W(n)$. Otherwise, the indicator function returns 0. In the latter case, we assume that an additional parameter that indicates the index of a forbidden window is also returned from the function . Lastly we denote by $\mathcal{I}(i)$ the $\lceil\log(n-\ell+1)\rceil$ bit representation of the index i.

Theorem 1. For a given $W(n)$-SA constraint $\mathcal{C}(n) \subseteq \Sigma^n$ such that $|\mathcal{C}(n)| \geq |S| = |\Sigma|^{n-1}$, and given

1) an injective function $\mathcal{R} : W(n) \to \Sigma^{\ell'}$,
2) an indicator $\mathbf{1}_{W(n)} : \Sigma^\ell \to \{0, 1\}$,

the encoder from Algorithm 1 is well-defined.

Proof: The algorithm converges when $\mathbf{1}_{\mathcal{C}(n)}(\boldsymbol{y}) = 1$, Therefore, all we need to prove is its convergence. To do so, we model the encoder as a walk on a directed graph $G = (V, E)$ with nodes representing string states and edges representing the iteration routine (Algorithm 1, Step 3). Let S represent the possible start nodes of the algorithm. Thus,

$$V = \Sigma^n, \quad S = \left\{ \boldsymbol{v} \circ 1 \;\middle|\; \boldsymbol{v} \in \Sigma^{n-1} \right\} \subseteq V,$$

$$E = \left\{ (\boldsymbol{u}, \boldsymbol{v}) \;\middle|\; \begin{array}{c} \boldsymbol{u} \notin \mathcal{C}(n), \\ \boldsymbol{v} = \boldsymbol{u}_1^{i-1} \circ \boldsymbol{u}_{i+\ell}^{n} \circ \mathcal{I}(i) \circ \mathcal{R}(\boldsymbol{u}_i^{i+\ell-1}) \circ 0 \end{array} \right\}.$$

Example of such a graph is shown in Figure 1. Let $\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, \ldots$ be the intermediate states of the encoder for some input $\boldsymbol{x} \in \Sigma^{n-1}$. Since $\boldsymbol{y}^{(i)} \in \Sigma^n$ for all $i \in \mathbb{N}$ and $|\Sigma^n| < \infty$, if the encoder does not converge, then there exist $i < j$ such that $\boldsymbol{y}^{(i)} = \boldsymbol{y}^{(j)}$. Therefore, we find that $\boldsymbol{y}^{(i)} \to \boldsymbol{y}^{(i+1)} \to \cdots \to \boldsymbol{y}^{(j-1)} \to \boldsymbol{y}^{(j)} = \boldsymbol{y}^{(i)}$ is a cycle

[1] Otherwise, the shorter words in $W(n)$ can be padded with all possible combinations until the maximal length.

Algorithm 1 Universal Iterative Encoder f

Input: $\boldsymbol{x} \in \Sigma^{n-1}$.
Output: $\boldsymbol{y} \in \mathcal{C}(n)$.
1: $\boldsymbol{y} \leftarrow \boldsymbol{x} \circ 1$.
2: **while** $\exists i$ s.t. $\boldsymbol{y}_i^{i+\ell-1} \in W(n)$ **do**
3: $\quad \boldsymbol{y} \leftarrow \boldsymbol{y}_1^{i-1} \circ \boldsymbol{y}_{i+\ell}^{n} \circ \mathcal{I}(i) \circ \mathcal{R}(\boldsymbol{y}_i^{i+\ell-1}) \circ 0$.
4: **end while**
5: **return** $\boldsymbol{y}$.

Algorithm 2 Universal Iterative Decoder g

Input: $\boldsymbol{y} \in \mathcal{C}(n)$ such that $f(\boldsymbol{x}) = \boldsymbol{y}$ for some $\boldsymbol{x} \in \Sigma^{n-1}$.
Output: $\boldsymbol{x} \in \Sigma^{n-1}$.
1: **while** $\boldsymbol{y}_n = 0$ **do**
2: $\quad i \leftarrow \mathcal{I}^{-1}(\boldsymbol{y}_{n-\ell+1}^{n-\ell+\lceil \log(n-\ell+1) \rceil})$.
3: $\quad \boldsymbol{y} \leftarrow \boldsymbol{y}_1^{i-1} \circ \mathcal{R}^{-1}(\boldsymbol{y}_{n-\ell+\lceil \log(n-\ell+1) \rceil+1}^{n-1}) \circ \boldsymbol{y}_i^{n-\ell}$.
4: **end while**
5: **return** $\boldsymbol{y}_1^{n-1}$.

in G. We note that $\boldsymbol{y}^{(i)}$ is reachable from a node in S as $\boldsymbol{y}^{(1)} \in S$ by properties of the encoder. Thus, we found a cycle C in G that is reachable from a node in S, yet this is impossible due to the fact that the in-degree of all nodes is at most one (as the encoder routine is injective) and that the in-degree of all nodes in S is zero (as the output of the routine always ends in 0).

■

Theorem 2. The decoder from Algorithm 2 is well-defined.

Proof: We prove that $g(f(\boldsymbol{x})) = \boldsymbol{x}$ for any $\boldsymbol{x} \in \Sigma^{n-1}$; let $\boldsymbol{x} \in \Sigma^{n-1}$ be given. Let $t(\boldsymbol{x})$ be the number of iterations performed in $f(\boldsymbol{x})$ (finite due to Theorem 1), and let $\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, \ldots, \boldsymbol{y}^{(t+1)}$ be the path taken by the encoder in G. Notice only $\boldsymbol{y}^{(1)} = \boldsymbol{x} \circ 1$ in S since the in-degree of all nodes in S is zero (as the output of the routine always ends in 0). By the design of Algorithm 2, we find that the decoder traverses the transpose graph G^T starting at $f(\boldsymbol{x}) = \boldsymbol{y}^{(t+1)}$ until a node in S is reached. Since the maximal in-degree in G is one, then we find that the maximal out-degree in G^T is also one; thus, the path taken by the decoder is well-defined and coincides with the reverse of the path taken by the encoder. Further, since only $\boldsymbol{y}^{(1)}$ is in S, the decoder terminates the while loop with $\boldsymbol{y}^{(1)} = \boldsymbol{x} \circ 1$ and returns $\boldsymbol{x}$. ■

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

Proof: Using the walk on G representation, and the fact that two paths generated by two distinct inputs are disjoint as nodes in G possess an in-degree of at most one, we can deduce that the sum of iterations the encoder does for all possible input is bounded by $|\Sigma^n \setminus S|$.

Therefore, we find that the average path length is

$$\frac{1}{|\Sigma^{n-1}|} \sum_{\boldsymbol{x} \in \Sigma^{n-1}} t(\boldsymbol{x}) \leq \frac{|\Sigma^n \setminus S|}{|\Sigma^{n-1}|} \leq \frac{|\Sigma^n|}{|\Sigma^{n-1}|} = |\Sigma| = O(1).$$

■

Corollary 1. The encoder possess $O(n \cdot T(\ell))$ average time complexity for $T(\ell)$ the maximal time amongst $\mathcal{R}, \mathbf{1}_{\mathcal{C}(n)}$.

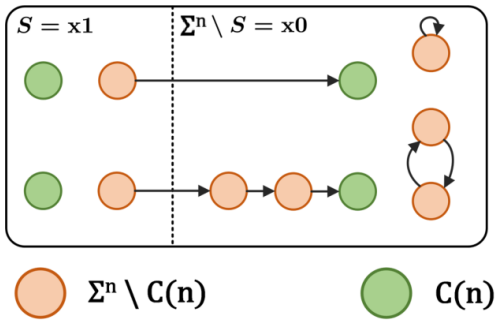

Figure 1. Example graph which Algorithm 1 traverses. The algorithm starts in S and applies the iteration routine until a valid node is reached (in $\mathcal{C}(n)$). While cycles exist, they are unreachable from S.

Corollary 2. The decoder possess $O(T(\ell)+n)$ average time complexity for $T(\ell)$ which is the time complexity of $\mathcal{R}^{-1}$.

While the implementation of $\mathbf{1}_{\mathcal{C}(n)}$ using $\mathbf{1}_{W(n)}$ results in $O(n \cdot \mathbf{1}_{W(n)}) = O(n \cdot \ell)$ time complexity, for specific constraints this can be improved to $O(n)$ as we demonstrate in [12]. Furthermore, it is always possible to use $O(\ell \cdot |W(n)|)$ space and achieve $T(\ell) = O(\ell \cdot \log(|W(n)|))$ time complexity using a binary search approach. Further, an efficient algorithm for computing $\mathcal{R}$ may be provided to reduce the overall complexity, as demonstrated in the following section.

IV. Constrained Windows

We study in this section a few of the common constrained codes presented and referenced in [1]: (1) the run-length-limited (RLL) constraint often utilized in magnetic and optical storage systems [17]–[19] , (2) the minimal Hamming weight constraint for energy-harvesting devices [20], [21], and (3) the balanced Hamming weight constraint for spectrum shaping [10], [22]–[25]. We apply the universal construction to each of these constraints and demonstrate state-of-the-art results with trivial adaptations to the core algorithm. We compare our construction to previous tailored methods in the different constraints, emphasizing the redundancy and parameter improvements of our constructions.

A. Run Length Limited

The ℓ-RLL constraint is a forbidden substring constraint that prohibits a substring of ℓ zeros,

$$W_{\ell-RLL}(n) = \{0^\ell\}. \tag{1}$$

Thus, we trivially define $\mathcal{R}(\boldsymbol{w}) = \varepsilon$. However, to ensure that appending the indices does not violate the constraint, in [6] the authors utilized a variation of the sequence replacement, taking the form $\boldsymbol{y} \leftarrow \boldsymbol{x}_1^{i-1} \circ \boldsymbol{x}_{i+\ell}^{n} \circ \mathcal{I}(i) \circ 10$. Notably, the number of distinct indices that may need encoding is $n-\ell+1$, but for simplicity, we use $\log(n) \geq \log(n-\ell+1)$ bits for index representation. Therefore, this method replaces ℓ bits with $\log(n)+2$ bits, making it viable for any $\ell \geq \log(n)+2$. Notice that [11] later established that the constraint preservation holds even without the variation, extending the applicability to $\ell \geq \log(n)+1$.

Notice that our general method, even without explicit knowledge of the proof from [11], converges to the same algorithm – as can be seen in the following construction.

Construction 1 ($\ell - RLL$ construction)**.** Using the universal iterative encoder and decoder, and the window representation $\mathcal{R}(\boldsymbol{w}) = \varepsilon$, we construct a valid encoder and decoder to the ℓ-RLL constraint. The construction uses one redundancy bit and is valid for $\ell \geq \log(n) + 1$.

B. Minimal Hamming Weight

A generalization of the ℓ-RLL constraint is introduced in [11] and is denoted as the minimal Hamming weight constraint – referenced as ℓ, p-MW. While ℓ-RLL restricts ℓ-substrings with a Hamming weight of zero, ℓ, p-MW broadens this constraint to encompass any weight less than $p(n)$.

$$W_{\ell,p-MW}(n) = \{\boldsymbol{w} \in \Sigma^{\ell} \mid w_H(\boldsymbol{w}) < p(n)\}. \quad (2)$$

The proposed window representation entails encoding each non-zero location with $\log(\ell)$ bits, achievable with at most $p(n) - 1$ such indices. In cases where the weight is smaller, the index value is set to be larger than the window size.

To ensure monotonic progression, [11] deviates from a one-bit redundancy approach, and append $p(n)$ consecutive ones in the algorithm initialization, resulting in more substantial redundancy. To underscore, employing our universal method yields a superior construction both in terms of the supported range and redundancy.

Construction 2 (ℓ, p-MW construction)**.** We design $\mathcal{R} : W(n) \to \Sigma^{\ell'}$ as follows: for $\boldsymbol{w} \in W(n)$, we define $\mathcal{R}(\boldsymbol{w}) = \mathcal{I}(i_1) \circ \mathcal{I}(i_2) \circ \ldots \circ \mathcal{I}(i_{p(n)-1})$, for $i_1, \ldots, i_{p(n)-1}$ the indices of the non-zero bits[2] in $\boldsymbol{w}$ and $\mathcal{I}(i)$ the binary encoding of i using $\lceil \log(\ell + 1) \rceil$ bits. Using the universal iterative encoder and decoder, and the window representation we construct a valid encoder and decoder to the ℓ, p-MW constraint. The construction uses one redundancy bit and is valid for $\ell \geq \lceil \log(n) \rceil + (p(n) - 1) \cdot \lceil \log(\ell + 1) \rceil + 1$.

C. Almost-Balanced Hamming Weight

An additional valuable extension of the minimal Hamming weight constraint is the almost-balanced (AB) Hamming weight, which stipulates that the Hamming weight of all ℓ-substrings falls within the interval $[p_1\ell, p_2\ell]$. Therefore the constrained-window set take the form,

$$W_{\ell,p_1,p_2-AB}(n) = \{\boldsymbol{w} \in \Sigma^{\ell} \mid w_H(\boldsymbol{w}) \notin [p_1\ell, p_2\ell]\}. \quad (3)$$

The previous state-of-the-art [7] establishes the existence of a one to one mapping $\mathcal{R}_{AB} : W_{\ell,p_1,p_2-AB}(n) \to \Sigma^{\ell'}$ for $\ell \geq \ln(n)/c^2$, where $c = \min\left\{\frac{1}{2} - p_1, p_2 - \frac{1}{2}\right\}$. However, this is limited to $\ell \geq \ln(n)/c^2 + 2$ due to the incorporation of additional bits required to ensure convergence. In contrast, our method allows $\ell \geq \ln(n)/c^2$ with one redundancy bit.

Construction 3 (ℓ, p_1, p_2-AB construction)**.** Using the universal iterative encoder and decoder, and the window representation $\mathcal{R}_{AB}$, we construct a valid encoder and decoder to the ℓ, p_1, p_2-AB constraint. The construction uses one redundancy bit and is valid for $n \geq 4, \ell \geq \frac{1}{c^2} \ln n$.

It is pertinent to note that the construction in [7] assumes the existence of mappings $\mathcal{R}_{AB}$ without explicitly determining them. Nevertheless, using our method, we present in concurrent work [16] an explicit construction for a sub-linear almost-balanced constraint. More precisely, the Hamming weight is in $\left[\frac{\ell}{2} - \alpha\sqrt{\ell}, \frac{\ell}{2} + \alpha\sqrt{\ell}\right]$, for $\alpha > \sqrt{\ln(2)}$.

[2]For $w_H(\boldsymbol{w}) < p(n) - 1$, we use a dummy index ℓ to indicate no entry.

D. Multiple Constraints

While previous methods are tailored to specific constraints, our general method allows us to combine multiple independent constraints and their respective representation into a unified construction that concurrently satisfies all constraints.

Theorem 3. Let $\mathcal{C}_1(n), \mathcal{C}_2(n), \ldots, \mathcal{C}_m(n) \subseteq \Sigma^n$ be given $W(n)$-SA constraints such that $\forall i, \left|\overline{\mathcal{C}_i(n)}\right| \leq |\Sigma|^{n-1-\lceil \log(m) \rceil}$, and

1) injective functions $\forall i, \mathcal{R}_i : W_i(n) \to \Sigma^{\ell' - \lceil \log(m) \rceil}$,
2) indicators $\forall i, \mathbf{1}_{W_i(n)} : \Sigma^{\ell} \to \{0, 1\}$.

There exists an efficient construction for $\mathcal{C}(n) = \bigcap_i \mathcal{C}_i(n)$ with one redundancy bit and $O(m \cdot T(n))$ average time complexity for $T(n)$ the maximal time complexity amongst $\mathcal{R}_1, \ldots, \mathcal{R}_m, \mathcal{R}_1^{-1}, \ldots, \mathcal{R}_m^{-1}$, and $\mathbf{1}_{\mathcal{C}_1(n)}, \ldots, \mathbf{1}_{\mathcal{C}_m(n)}$.

Proof: We will demonstrate the desired construction as a special case of Theorem 1.

1) We verify that $\left|\overline{\mathcal{C}(n)}\right| \leq \Sigma^{n-1}$ as follows,

$$\begin{aligned}\left|\overline{\mathcal{C}(n)}\right| = \left|\bigcup_i \overline{\mathcal{C}_i(n)}\right| &\leq \sum_i \left|\overline{\mathcal{C}_i(n)}\right| \\ &\leq m \cdot |\Sigma|^{n-1-\lceil \log(m) \rceil} \leq |\Sigma|^{n-1}.\end{aligned}$$

2) We propose the following injective function $\mathcal{R} : W(n) \to \Sigma^{\ell'}$, where $\mathcal{R}(\boldsymbol{w})$ is computed as follows: let i be the minimal index such that $\boldsymbol{w} \in W_i(n)$ (exists since $\boldsymbol{x} \in \overline{\mathcal{C}(n)}$), then $\mathcal{R}(\boldsymbol{w}) = \mathcal{R}_i(\boldsymbol{w}) \circ \mathcal{I}(i)$, where $\mathcal{I}(i)$ is the encoding of i using $\lceil \log(m) \rceil$ bits.
3) An indicator function $\mathbf{1}_{\mathcal{C}(n)} : \Sigma^n \to \{0, 1\}$ is computed as $\mathbf{1}_{\mathcal{C}(n)}(\boldsymbol{x}) = \bigwedge_i \mathbf{1}_{\mathcal{C}_i(n)}(\boldsymbol{x})$.

Therefore, according to the correctness of Theorem 1, we find that there exists a construction for $\mathcal{C}(n)$ with $r = 1$ and $O(m \cdot T(n))$ average time complexity for $T(n)$ the maximal time complexity amongst $\mathcal{R}_1, \ldots, \mathcal{R}_m, \mathcal{R}_1^{-1}, \ldots, \mathcal{R}_m^{-1}$, and $\mathbf{1}_{\mathcal{C}_1(n)}, \ldots, \mathbf{1}_{\mathcal{C}_m(n)}$. ∎

V. Substring-Pair Constraints

In recent years, studies have underscored the necessity for constraints that fall outside the conventional definition of substring-avoiding constraints but can still be addressed with the sequence replacement method. These constraints deviate from a predefined list of forbidden windows $W(n)$, and instead, windows are restricted by more sophisticated conditions. The most commonly used family of constraints involves conditions on pairs of substrings. For example, the repeat-free constraint [13], which requires the uniqueness of any $\ell(n)$-substring $\boldsymbol{w}$ in $\boldsymbol{x}$. The sequence-replacement technique can tackle the repeat-free constraint by searching for duplicate substrings and then replacing one of them with an appended encoding of both their indices.

A. Overlapping Pairs

While addressing pairs of windows, it is crucial to acknowledge scenarios involving an intersection of these windows since then the removal of a duplicate substring can also remove parts of the original substring. Conversely, this also guarantees that the suffix of the substring is a repetition of its prefix (since the suffix of the original was identical to the prefix of the duplicate) and thus the entire substring can be

reconstructed only from its prefix. The identity relationship between the two substrings can be generalized to any bitwise function f. We define the ℓ-bitwise-overlapping-pairs (OP) constraints as,

$$\mathcal{C}_{\ell-OP}(n) = \left\{ \boldsymbol{x} \in \Sigma^n \;\middle|\; \begin{array}{c} \nexists 1 \le i < j \le n-\ell(n)+1 \;: \\ \boldsymbol{x}_i^{i+\ell(n)-1} = f(\boldsymbol{x}_j^{j+\ell(n)-1}) \end{array} \right\}, \tag{4}$$

Construction 4 ($\ell - OP$ construction)**.** Using the universal iterative encoder and decoder, and the window representation $\mathcal{R}(\boldsymbol{w}, \boldsymbol{u}) = \mathcal{I}(\boldsymbol{u})$ (where $\mathcal{I}(\boldsymbol{u})$ is the bit representation index of the location of $\boldsymbol{u}$), we construct a valid encoder and decoder to the ℓ-OP constraint. The construction uses one redundancy bit and is valid for $\ell \ge 2\log(n) + 1$.

The state-of-the-art construction for the ℓ-repeat-free (ℓ-RF) constraint was given in [13], where the authors addressed $\ell \ge 2\log n + 2$ and utilized 2 bits of redundancy. Using our universal construction yields improvements in both achievable range and redundancy.

B. Non-Overlapping Pairs

In cases where the constraint does not consider overlapping windows, we can expand our construction to any function $f : \Sigma^\ell \to \Sigma^\ell$. We define the ℓ-non-overlapping-pairs (NP) constraints as,

$$\mathcal{C}_{\ell-NP}(n) = \left\{ \boldsymbol{x} \in \Sigma^n \;\middle|\; \begin{array}{c} \nexists 1 \le i < n-\ell+1, \\ i+\ell \le j \le n-\ell+1 \;: \\ f(\boldsymbol{x}_i^{i+\ell-1}) = \boldsymbol{x}_j^{j+\ell-1} \end{array} \right\}. \tag{5}$$

Construction 5 ($\ell - NP$ construction)**.** Using the universal iterative encoder and decoder, and the window representation $\mathcal{R}(\boldsymbol{w}, \boldsymbol{u}) = \mathcal{I}(\boldsymbol{u})$, we construct a valid encoder and decoder for the ℓ-NP constraint. The construction uses one redundancy bit and is valid for $\ell \ge 2\log(n) + 1$.

An important application of this constraint is found in the secondary structure problem [14], where the objective is to prevent a DNA sequence from folding back upon itself. In this context, a word is considered ℓ-secondary structure avoidance (ℓ-SSA) if there is no pair of *non-overlapping* substrings $\boldsymbol{w}, \boldsymbol{u}$ of length ℓ such that $\boldsymbol{w} = RC(\boldsymbol{u})$, where the reverse complement function RC is defined as $\boldsymbol{w}_i = \bar{\boldsymbol{u}}_{\ell-i}$.

The previous construction of [14] is applicable for $\ell \ge 6\log n + 4$, yet it may be complex to extend beyond the binary alphabet. The recent work [26] improves this result by relaxing the constraint. Nonetheless, applying our universal encoder results in a better achievable range without the need to modify the constraint.

VI. Generalized Constructions

We summarize our contributions regarding general parametric constraints with arbitrary redundancy bits. These results are easily extended to non-binary alphabet. The proofs are based on a similar graph reduction, and due to space limitation, they are omitted. The details can be found in the extended version of this work [15].

We begin with a construction for any parametric constraint $\mathcal{C}(n)$ using a single redundancy bit,

Construction 6. Let $\mathcal{C}(n) \subseteq \Sigma^n$ be a given constraint such that $\left|\overline{\mathcal{C}(n)}\right| \le |\Sigma|^{n-1}$, and given

1) an injective function $\xi : \overline{\mathcal{C}(n)} \to \Sigma^{n-1}$,
2) an indicator $\mathbf{1}_{\mathcal{C}(n)} : \Sigma^n \to \{0, 1\}$,

there exists an efficient construction with a single redundancy bit and $O(T(n))$ average time complexity for $T(n)$ the maximal time complexity amongst ξ, ξ^{-1} and $\mathbf{1}_{\mathcal{C}(n)}$.

To address scenarios involving more than a single redundancy bit, we introduce an embedding function ψ, and the set S indicator $\mathbf{1}_S$.

Construction 7 (Universal)**.** For a given constraint $\mathcal{C}(n) \subseteq \Sigma^n$, and given

1) a subset $S \subseteq \Sigma^n$ such that $|\mathcal{C}(n)| \ge |S| \ge |\Sigma|^k$,
2) an injective embedding $\psi : \Sigma^k \to S$ and indicator $\mathbf{1}_S : \Sigma^n \to \{0, 1\}$,
3) an injective function $\varphi : \overline{\mathcal{C}(n)} \to \overline{S}$ and indicator $\mathbf{1}_{\mathcal{C}(n)} : \Sigma^n \to \{0, 1\}$,

there exists an efficient construction with $r = n - k$ redundancy symbols and $O(|\Sigma|^r \cdot T(n))$ average time complexity for $T(n)$ the maximal complexity amongst $\varphi, \varphi^{-1}, \mathbf{1}_S, \mathbf{1}_{\mathcal{C}(n)}$.

Note that our explicit constructions are using $\psi : \Sigma^{n-1} \to S$ such that $\psi(\boldsymbol{x}) = \boldsymbol{x} \circ 1$, and $\mathbf{1}_S(\boldsymbol{x}) = x_n$ for $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$.

VII. Conclusion and Discussion

Parametric constrained coding generalizes the well-known field of fixed constrained coding to address constraints that depend on the message length and portray some general property (e.g., avoid windows of $\log(n)$ consecutive zeros). While there is a universal solution to any *fixed* constrained coding task, the lack of such a solution for *parametric* constraints has led to custom algorithms being designed for each constraint – typically following the general guideline of the sequence-replacement concept. Specifically, most approaches are based on iteratively replacing invalid substrings in the transmitted message, yet they require complex custom transition functions that enable the monotonic progression which guarantees convergence. In this work, we propose a novel universal framework for all parametric constrained coding problems that can generalize to all constraints due to a surprising proof that demonstrates that there is no need for monotonic progression to guarantee efficient convergence. We apply this framework to several problems (see Table I) to provide state-of-the-art results with minimal effort.

Future research should address the diverse challenges posed by different parametric constraints. Several possible directions and open problems are listed below.

1) Study the capacity and achievable parameters of different parametric constrained channels and compare them to the constructions obtained by our framework.
2) Analyze the worse-case time complexity of the proposed constructions.
3) Find an explicit and efficient embedding function ψ that can be used for redundancy of more than one symbol (see Construction 7).

REFERENCES

[1] K. A. S. Immink, "Innovation in constrained codes," *IEEE Communications Magazine*, vol. 60, no. 10, pp. 20–24, 2022.

[2] B. H. Marcus, R. M. Roth, and P. H. Siegel, "An introduction to coding for constrained systems," *Lecture notes*, 2001.

[3] R. Adler, D. Coppersmith, and M. Hassner, "Algorithms for sliding block codes-an application of symbolic dynamics to information theory," *IEEE Transactions on Information Theory*, vol. 29, no. 1, pp. 5–22, 1983.

[4] A. Van Wijngaarden and K. S. Immink, "On the construction of constrained codes employing sequence replacement techniques," in *Proceedings of IEEE International Symposium on Information Theory*. IEEE, 1997, p. 144.

[5] A. J. d. L. van Wijngaarden, "Frame synchronization techniques: Rahmensynchronisationsverfahren," Ph.D. dissertation, 1998.

[6] A. J. Van Wijngaarden and K. A. S. Immink, "Construction of maximum run-length limited codes using sequence replacement techniques," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 2, pp. 200–207, 2010.

[7] 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.

[8] C. Schoeny, A. Wachter-Zeh, R. Gabrys, and E. Yaakobi, "Codes correcting a burst of deletions or insertions," *IEEE Transactions on Information Theory*, vol. 63, no. 4, pp. 1971–1985, 2017.

[9] K. A. S. Immink and K. Cai, "Design of capacity-approaching constrained codes for dna-based storage systems," *IEEE Communications Letters*, vol. 22, no. 2, pp. 224–227, 2017.

[10] ——, "Properties and constructions of constrained codes for DNA-based data storage," *IEEE Access*, vol. 8, pp. 49 523–49 531, 2020.

[11] M. Levy and E. Yaakobi, "Mutually uncorrelated codes for DNA storage," *IEEE Transactions on Information Theory*, vol. 65, no. 6, pp. 3671–3691, 2018.

[12] 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.

[13] O. Elishco, R. Gabrys, E. Yaakobi, and M. Médard, "Repeat-free codes," *IEEE Transactions on Information Theory*, vol. 67, no. 9, pp. 5749–5764, 2021.

[14] T. T. Nguyen, K. Cai, H. M. Kiah, D. T. Dao, and K. A. S. Immink, "On the design of codes for DNA computing: Secondary structure avoidance codes," *arXiv preprint arXiv:2302.13714*, 2023.

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

[16] ——, "Optimal almost-balanced sequences," in *Proceedings of the IEEE International Symposium on Information Theory (ISIT)*, 2024.

[17] K. S. Immink, "Runlength-limited sequences," *Proceedings of the IEEE*, vol. 78, no. 11, pp. 1745–1759, 1990.

[18] K. A. S. Immink, *Codes for mass data storage systems*. Shannon Foundation Publisher, 2004.

[19] ——, "A survey of codes for optical disk recording," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 4, pp. 756–764, 2001.

[20] A. Tandon, H. M. Kiah, and M. Motani, "Bounds on the size and asymptotic rate of subblock-constrained codes," *IEEE Transactions on Information Theory*, vol. 64, no. 10, pp. 6604–6619, 2018.

[21] E. Rosnes, Á. I. Barbero, and Ø. Ytrehus, "Coding for inductively coupled channels," *IEEE Transactions on Information Theory*, vol. 58, no. 8, pp. 5418–5436, 2012.

[22] F.-L. Chang, W.-W. Hu, D.-H. Lee, and C.-T. Yu, "Design and implementation of anti low-frequency noise in visible light communications," in *International Conference on Applied System Innovation (ICASI)*. IEEE, 2017, pp. 1536–1538.

[23] J. Franklin and J. Pierce, "Spectra and efficiency of binary codes without DC," *IEEE Transactions on Communications*, vol. 20, no. 6, pp. 1182–1184, 1972.

[24] O. P. Babalola and V. Balyan, "Efficient channel coding for dimmable visible light communications system," *IEEE Access*, vol. 8, pp. 215 100–215 106, 2020.

[25] D. T. Dao, H. M. Kiah, and T. T. Nguyen, "Average redundancy of variable-length balancing schemes à la knuth," *arXiv preprint arXiv:2204.13831*, 2022.

[26] R. Zhang and H. Wu, "On secondary structure avoidance of codes for DNA storage," *Computational and Structural Biotechnology Journal*, 2023.