Abstract—Reliability is an inherent challenge for the emerging nonvolatile technology of racetrack memories, and there exists a fundamental relationship between codes designed for racetrack memories and codes with constrained periodicity. Previous works have sought to construct codes that avoid periodicity in windows, yet have either only provided existence proofs or required high redundancy. This paper provides the first constructions for avoiding periodicity that are both efficient (average-linear time) and with low redundancy (near the lower bound). The proposed algorithms are based on iteratively repairing windows which contain periodicity until all the windows are valid. Intuitively, such algorithms should not converge as there is no monotonic progression; yet, we prove convergence with average-linear time complexity by exploiting subtle properties of the encoder. Overall, we both provide constructions that avoid periodicity in all windows, and we also study the cardinality of such constraints.

I. INTRODUCTION

Racetrack memories are an emerging form of nonvolatile memory with extremely high density that have the potential to overcome the fundamental constraints of traditional memory devices [1], [2]. They consist of magnetic nanowires that store numerous bits through magnetic polarity; their value is accessed by shifting the bits stored in each wire to heads at fixed locations. Unfortunately, the shift operation is highly unreliable, thereby leading to position errors in the form of deletions and sticky insertions [3]. Codes that address these errors have a fundamental relationship to constrained periodicity due to the reading structure involving multiple heads simultaneously reading at fixed offsets [4], [5].

This paper aims to develop efficient codes that constrain periodicity in all windows of encoded messages. Specifically, we consider both the \( l \)-window \( p \)-period avoidance (PA) constraint where all windows of length \( l \) cannot contain a period \( p \), and the \( l \)-window \( p \)-least-period avoiding (LPA) constraint where all windows of length \( l \) cannot contain any period \( p' < p \). These constraints were first considered by Chee et al. [4], where a lower bound on the cardinality proved the existence of a binary LPA code with \( l = \lceil \log(n) \rceil + p + 1 \) using a single redundancy symbol. Sima and Bruck [6] later proposed an \( O(n^2 \log n) \) time algorithm for the constraint \( l = \lceil \log(n) \rceil + 3p - 2 \) using \( p + 1 \) redundancy symbols; yet, there remains a significant gap in the redundancy between this explicit construction and the lower bound provided by Chee et al. [4]. Conversely, in this paper, we propose an \( O(n) \) average-time construction of LPA codes using a single redundancy symbol for \( l \) being the minimal integer satisfying \( l = \lceil \log(n - l + 2) \rceil + p + 1 \). Further, we prove that LPA codes that use a single redundancy symbol exist only for values of \( l \) that satisfy \( l \geq \log(n - 2l + p) + p - 3.5 \).

The proposed approach is based on iteratively repairing invalid windows until a legal message is encountered. Specifically, as long as there exists a window with invalid periodicity, we remove the window and append an alternate encoding of the window (of identical length). While intuitively this algorithm should not converge due to the lack of monotonic progression (e.g., appended symbols may create additional periodic windows) and the existence of cycles (e.g., repairing an invalid window may lead to the original message), we show that subtle properties of the algorithm guarantee convergence. Further, we prove that only \( O(1) \) windows are repaired on average, leading to \( O(n) \) average time encoding and decoding.

This paper is organized as follows. Section II begins by providing background on periodicity, the previously-proposed codes, and the run-length-limited (RLL) constraint. Section III presents the proposed construction, and Section IV explores several generalizations. Section V provides a cardinality analysis, and finally Section VI concludes this paper. Some proofs are omitted and are available in the extended version [7].

II. DEFINITIONS, PRELIMINARIES, AND RELATED WORKS

We begin with several definitions and various results in the theory of periodicity, continue with the previous works for periodicity-constrained codes [4], [6], and conclude with background on the run-length-limited (RLL) constraint [8].

A. Notations

For all \( i \), we denote \([i] = \{k \in \mathbb{N} \mid 1 \leq k \leq i\}\). Let \( \Sigma_q \) be a finite alphabet of size \( q \) and \( \Sigma_q^n \) be the set of all vectors of length \( n \) over \( \Sigma_q \); without loss of generality, \( 0, 1 \in \Sigma_q \). For a subset \( A \subseteq \Sigma_q^n \), we denote by \( \overline{A} \) the complement of \( A \). For a vector \( s = (s_1, \ldots, s_n) \) and \( i, j \in [n], i \leq j \), we denote by \( s_i^j \) the window \((s_i, \ldots, s_j)\) of \( s \). A zero run of length \( r \) of a vector \( s \in \Sigma_q^n \) is a window \( s_i^{i+r-1} \), \( i \in [n-r+1] \) such that \( s_i = \cdots = s_{i+r-1} = 0 \). The notation \( st \) denotes the concatenation of the two vectors \( s \) and \( t \), and \( sk^k \) denotes the concatenation of \( s \) with itself \( k \) times. Unless stated otherwise, \( \log \) refers to \( \log_q \), where \( q \) is the size of the alphabet.

B. Periodicity Definitions

We begin this subsection with a definition for the periodicity of a vector, continue by defining a periodicity-avoiding (PA) vector which avoids a specific period in all windows, and then extend this to a least-periodicity-avoiding (LPA) vector which avoids all periods up to a specific value in all windows.
Definition 1. For $s \in \Sigma_q^n$, $p \in [n-1]$ is called a period of $s$ if for all $i \in [n-p]$, $s_i = s_{i+p}$.

Definition 2 (PA). For $s \in \Sigma_q^n$, $s$ is an $\ell$-window $p$-period avoiding vector if every window of length $\ell$ does not possess period $p$. Let $B_q(n, \ell, p)$ be the set of such vectors, and let $b_q(n, \ell, p) = |B_q(n, \ell, p)|$.

Definition 3 (LPA). For $s \in \Sigma_q^n$, $s$ is an $\ell$-window $p$-least-period avoiding vector if $s$ is an $\ell$-window $p'$-period avoiding (PA) vector for all $p' < p$. Equivalently, every window of length $\ell$ in $s$ does not contain any period $p' < p$. Let $A_q(n, \ell, p)$ be the set of all such vectors $s$, and let $a_q(n, \ell, p) = |A_q(n, \ell, p)|$. Notice that $A_q(n, \ell, p) = \bigcap_{p'=1}^{p-1} B_q(n, \ell, p)$ as multiples of periods are periods. A code $C$ is called an $(\ell, p)$-LPA code if $C \subseteq A_q(n, \ell, p)$. If the values of $\ell$ and $p$ are clear from the context, it is simply referred to as an LPA code.

This paper tackles the following three problems:

Problem 1. Design $\ell$-window $p$-period LPA codes with efficient encoding/decoding that minimize the value of $\ell$ while requiring only a single redundancy symbol.

Problem 2. Design $\ell$-window $p$-period LPA codes with efficient encoding/decoding that minimize the number of redundancy symbols for a given small value of $\ell$.

Problem 3. Study the values of $a_q(n, \ell, p)$ and $b_q(n, \ell, p)$.

C. Theory of Periodicity

Periodicity has been widely explored as a theoretical concept; we highlight key theorems utilized in Sections IV and V.

Theorem 1 (Fine and Wilf’s [9], [10]). Let $s \in \Sigma_q^n$ with periods $p_s$ and $p_t$ where $n \geq p_s + p_t - \gcd(p_s, p_t)$. Then $\gcd(p_s, p_t)$ is also a period of $s$.

Theorem 1 provides conditions for the uniqueness of a period in a message: if there are two periods $p_s, p_t < \lceil n/2 \rceil + 2$, then $p_s$ and $p_t$ are both multiples of a smaller period ($\gcd(p_s, p_t)$). Therefore, by extending a message with a symbol that contradicts the minimal period, we find:

Corollary 1. Let $s \in \Sigma_q^n$. Then there exists $a \in \Sigma_q^1$ such that $sa \in \Sigma_q^{n+1}$ contains no periods less than $\lceil n/2 \rceil + 2$.

D. Related Works on Constrained Periodicity

Problem 1 was first considered by Chee et al. [4], which presented a lower bound on $a_q(n, \ell, p)$ to prove that an LPA code with $\ell = \lceil \log(n) \rceil + p + 1$ and a single redundancy symbol exists; yet, an explicit construction was not derived. Sima and Bruck [6] later proposed an explicit construction with $O(n^2p\log n)$ time complexity for $\ell = \lceil \log(n) \rceil + 3p - 2$ using $p + 1$ redundancy symbols; yet, the redundancy is significantly higher than Chee et al. [4]. Section V highlights the main results from Chee et al. [4] regarding LPA codes, including the lower bound on $a_q(n, \ell, p)$ and a relationship between the PA constraint and the run-length-limited (RLL) [8] constraint.

E. The Run-Length-Limited (RLL) Constraint

The run-length-limited (RLL) constraint restricts the length of runs of consecutive zeros within encoded messages [8], [11]. Similar to [8], we consider the $(0, k)$-RLL constraint, which imposes the length of every run of zeros to be at most $k$, and for simplicity refer to this constraint as the $k$-RLL constraint. Below is the definition of the constraint and the state-of-the-art construction for a single redundancy symbol.

Definition 4 (RLL). A vector $s \in \Sigma_q^n$ satisfies the $k$-RLL constraint if there are no zero runs of length $k$. Let $R_q(n, k)$ be the set of such vectors, and let $r_q(n, k) = |R_q(n, k)|$. A code satisfying the $k$-RLL constraint is called a $k$-RLL code.

Construction 1 ([8]). For all $n$ and $k = \lceil \log(n) \rceil + 1$, there exists an explicit construction of $k$-RLL codes with a single redundancy symbol and encoding/decoding with $O(n)$ time.

III. SINGLE-SYMBOL LPA CONSTRUCTION

This section tackles Problem 1 through an approach of iteratively repairing invalid windows in the vectors, resulting in the following construction for a single redundancy symbol.

Construction 2. There exists an explicit construction of $(\ell, p)$-LPA codes for $\ell$ being the minimal value satisfying $\ell = \lceil \log(n - \ell + 2) \rceil + p + 1$, a single redundancy symbol, and $O(n)$ average-time encoding and decoding complexity.

The main idea is for the encoder to iteratively repair invalid windows until no such windows exist, and then reverse these steps in the decoder. While this is relatively simple, the difficulty remains in proving its convergence due to the lack of monotonic progression: repairing a certain window may cause other previously-valid windows (both to the left and the right) to become invalid. Surprisingly, through a reduction to an acyclic graph walk, we nonetheless show that subtle properties of the repair routine inherently guarantee convergence.

This section continues by detailing the proposed encoder and decoder algorithms, proving their convergence through a reduction to an acyclic graph walk, and attaining $O(n)$ average time complexity. For the remainder of this section, $\ell$ is the minimal integer that satisfies $\ell = \lceil \log(n - \ell + 2) \rceil + p + 1$.

A. Proposed Encoder and Decoder

The encoder, which is explicitly described in Algorithms 1 and 2, iteratively removes invalid windows while appending a representation of the steps performed to the message. Inspired by Construction 1, the redundancy symbol encodes whether any steps were taken: the symbol is initialized to one at the start, and becomes zero if a repair step is taken. The representation of a single step encodes the kernel of the periodic window removed (the first $p'$ symbols in a window with periodicity $p'$), the periodicity ($p'$), and the index of the window. Both the kernel and $p'$ are encoded within the same $p$ symbols by appending a one padded with zeros to the kernel. Notice that the message size is unchanged as $\ell$ was chosen to satisfy $\ell = \lceil \log(n - \ell + 2) \rceil + p + 1$. Overall, we proceed with such repair steps until there exists no invalid window.
Algorithms 1 and 2 perform the following steps:

Input: $x \in \Sigma_q^*$. 
Output: $y \in \Sigma_q^{n+1}$ such that $y \in A_q(n+1, \ell, p)$. 
1: $y \leftarrow x1$
2: while $y \notin A_q(n+1, \ell, p)$ do
3: 
   $y \leftarrow \text{Repair}(y)$.
4: end while
5: return $y$.

Algorithm 2 Repair 

Input: $y \in \Sigma_q^{n+1}$ such that $y \notin A_q(n+1, \ell, p)$. 
Output: $y \in \Sigma_q^{n+1}$ such that $y_{n+1} = 0$. 
1: $i \leftarrow$ index of first $\ell$-window in $y$ with period $p' < p$.
2: Append $y_i^{i+p'-1}10^{p-p'-1}$ to the end of $y$ (i.e., $y_i^{i+p'-1}$, then one, then $p-p'-1$ zeros).
3: Append the representation of $i$ (using $[\log(n-\ell+2)]$ symbols; zero-indexed) to $y$.
4: Append $0$ to $y$.
5: Remove the $\ell$-window at index $i$.
6: return $y$

The decoder reverses the steps of the encoder, as inspired by the encoder from Construction 1. The redundancy symbol is utilized to determine whether the last symbols of the message encode a step that was performed by the encoder. If so, then the decoder removes the step representation, reconstructs the invalid window by extending the given kernel according to the given period, and inserts it at the given index.

Example 1 exemplifies the encoder for the binary case.

Example 1. Let $n = 14$ and $p = 4$ (thus $\ell = 8$) with 
\[ x = 10001010101100. \]

Algorithms 1 and 2 perform the following steps:
1) $y = x1 = 100010101011001$.
2) $y \leftarrow \text{Repair}(y)$.
   2.1. The 8-window starting at $i = 3$ (01010101) is invalid as it possesses period $p' = 2 < p$.
   2.2. $y = y01101^1 = 100010101011001 0110$.
   2.3. $y = y011 = 100010101011001 0110 011$.
   2.4. $y = y0 = 100010101011001 0110 011 0$. 
   2.5. Remove the 8-window at index $i = 3$ from $y$.
   2.6. Return $y = 10010010101100$. 
3) $y \leftarrow \text{Repair}(y)$.
   3.1. The 8-window starting at $i = 0$ (10010010) is invalid as it possesses period $p' = 3 < p$.
   3.2. $y = y00^1 = 10010010101101011001$.
   3.3. $y = y000 = 10010010101101011001 000$. 
   3.4. $y = y0 = 10010010101101011001 000 0$. 
   3.5. Remove the 8-window at index $i = 0$ from $y$.
   3.6. Return $y = 11001101010100$. 
4) Return $y = 11001101010100000 \in A_q(15, 8, 4)$. 

Notice that the first call to the Repair function in Example 1 created the invalid window which was then addressed by the second call. That is, while Repair may fix the current window, it may also create other invalid windows. Therefore, it is unclear whether the algorithm will ever converge, considering that each repair may lead to additional invalid windows. Indeed, we find that there even exist states ($y \in \Sigma_q^{n+1}$) that if ever reached will cause Algorithm 1 to never converge. This scenario is demonstrated in the following example.

Example 2. Let $n = 14$ and $p = 4$ (thus $\ell = 8$), with 
\[ y = 111111010101010. \]

The repair routine (Algorithm 2) would perform the following steps if $y$ is reached by Algorithm 1 as an intermediate state:
1) The window starting at $i = 5$ (101010101) is invalid as it possesses period $p' = 2 < p$.
2) $y = y1010^1 = 11111101010101010 1010$.
3) $y = y01 = 11111101010101010 1010 101$. 
4) $y = y0 = 11111101010101010 1010 101 0$. 
5) Remove window at index $i = 5$ from $y$.
6) Return $y = 111111010101010$. 

That is, Repair($y$) = $y$. Therefore, if Algorithm 1 were to ever reach this $y$, then the encoder would never converge.

Nonetheless, as proven in Section III-B, the proposed encoder always converges as it inherently avoids such intermediate states (e.g., Example 2) due to subtle properties of the Repair function. Further, Section III-C demonstrates that the number of steps taken is only $q - 1 = O(1)$ on average; thus, the encoder and decoder time complexity is $O(n)$ on average.

B. Convergence Proof

This section proves the convergence of the proposed encoder and decoder through a reduction to an acyclic graph walk. We show that the encoder inherently avoids intermediate states that will lead to infinite loops (e.g., Example 2) by exploiting two subtle properties of the Repair function: the fact that it is injective, and the fact that Repair($y$) always ends with zero. The intuition for the proof is as follows. Let $y$ be given such that $\text{Repair}(y) = y$ (as in Example 2), we show that the encoder will never reach such $y$ as an intermediate state. Since $\text{Repair}(y) = y$, then $y$ ends with zero; thus, the encoder will never start the repair steps with $y$. Further, as $\text{Repair}$ is injective, then there exists no $z \neq y$ such that $\text{Repair}(z) = y = \text{Repair}(y)$; thus, $y$ cannot be reached from a different intermediate state $z$. Therefore, the encoder will never reach any such $y$ as the encoder cannot start with such $y$ and the encoder will never update the intermediate state to be such $y$.

We generalize the above intuition in Theorem 2 to also address cyclic structures that consist of more than one intermediate state (e.g., $\text{Repair}(y_1) = y_2$ and $\text{Repair}(y_2) = y_1$).

Lemma 1. The Repair function from Algorithm 2 is injective (that is, for all $z \neq y$, it holds that $\text{Repair}(z) \neq \text{Repair}(y)$).

Theorem 2. The encoder from Algorithm 1 is well-defined.

Proof: Notice that if the encoder converges, then the output is in $A_q(n+1, \ell, p)$ by design, and thus a valid message is returned. The difficulty remains in proving the convergence.
Theorem 3. The decoder is well-defined and correct.

Proof: The proof is similar to the proof of Construction 1.

C. Time Complexity

This section extends the analysis of Section III-B to demonstrate that the average time complexity of the encoder and decoder is $O(n)$. We first show that the average number of steps is $O(1)$, and then propose an $O(n)$ algorithm for each step (i.e., the repair and inverse-repair functions).

Lemma 2. The average number of iterations of the while loop in Algorithm 1 is at most $q - 1 = O(1)$.

Proof: As shown in Theorem 2, an execution of Algorithm 1 is equivalent to a walk on $G$. We notice that the two paths generated by two distinct inputs are disjoint as nodes in $G$ possess an in-degree of at most one. Thus, the sum of the lengths of all paths is the size of the union of all paths from all possible inputs. Therefore, as all paths are contained in $V \setminus S$ (excluding start nodes), the sum is bounded by $|V \setminus S|$. Let $t(x)$ be the length of the path for input $x \in \Sigma_q^n$: we find,

$$
\sum_{x \in \Sigma_q^n} t(x) \leq |V \setminus S| = q^{n+1} - q^n = (q - 1) \cdot q^n.
$$

Therefore, we find that the average path length is $q - 1 = O(1)$,

$$
\frac{1}{q^n} \sum_{x \in \Sigma_q^n} t(x) \leq \frac{(q - 1) \cdot q^n}{q^n} = q - 1 = O(1).
$$

Corollary 2. The encoder possesses $O(n)$ average time for $\ell \geq 2p - 2$ and $O(np)$ average time otherwise.

Corollary 3. The decoder possesses $O(n)$ average time.

IV. EXTENSIONS OF THE LPA ENCODER

This section tackles Problem 2 by proposing generalizations of Construction 2 to support smaller window sizes $(\ell < \lceil \log(n - \ell + 2) \rceil + p + 1)$ while minimizing the number of redundancy symbols. We demonstrate a trade-off between three proposed constructions which are all based on partitioning the input message into independent segments.

Construction 3. For any given $n, \ell, p$, there exists an explicit construction for $(\ell, p)$-LPA codes with $k$ redundancy symbols, for $k$ the smallest integer such that $\ell \geq 2 \cdot (\lceil \log(n/k - \ell/2 + 2) \rceil + p + 1)$, and $O(n)$ average-time encoding/decoding.

Construction 4. For given $n, \ell, p$ such that $\ell \geq 3p - 3$, there exists an explicit construction for $(\ell, p)$-LPA codes with $(p + 3) \cdot (k - 1) + 1$ redundancy symbols, where $k$ is the smallest value that satisfies $\ell \geq \lceil \log(n/k - \ell + 2) \rceil + p + 1$, and $O(n)$ average-time encoding/decoding.

Construction 5. For given $n, \ell, p$ such that $\ell \geq 4p - 7$, there exists an explicit construction for $(\ell, p)$-LPA codes with $3 \cdot k - 2$ symbols of redundancy, where $k$ is the smallest value that satisfies $\ell = \lceil \log(n/k - \ell + 2) \rceil + p + 1$, and $O(n)$ average-time encoding/decoding.

Overall, for given $n, \ell, p$, we seek the construction with minimal redundancy. First, note that Construction 5 is preferable over Construction 4 when $\ell \geq 4p - 7$. Further, Construction 4 is preferable over Construction 3 when $\ell \geq 3p - 3$ and

$$
q^{\ell/2-p-1} + \frac{\ell}{2} - 2 < \frac{q^{\ell-p-1} + \ell - 2}{p + 3}.
$$

Similarly, Construction 5 requires less redundancy than Construction 3 when $\ell \geq 4p - 7$ and

$$
q^{\ell/2-p-1} + \frac{\ell}{2} - 2 < \frac{q^{\ell-p-1} + \ell - 2}{3}.
$$
V. CARDINALLY ANALYSIS

This section analyzes the cardinality of the PA and LPA constraints, extending the analysis of Chee et al. [4]. We begin with the first upper bound for $a_q(n, \ell, p)$ and a demonstration that $\ell = \log(n - 2\ell + p) + p - 3.5$ is a lower bound for single-symbol redundancy. We continue with several interesting exact formulas for the remaining cases not covered by the bounds.

A. LOWER AND UPPER BOUNDS ON $a_q(n, \ell, p)$

We begin with results from [4] in Theorems 4 and 5, and then propose additional bounds via an RLL reduction.

**Theorem 4 (Chee et al. [4]).** For all $n, \ell, p$, and for all $q$,
\[
a_q(n, \ell, p) \geq q^n \cdot \left(1 - \frac{n}{(q-1)^2} \cdot q^{\ell-p}\right).
\]

In particular, for $\ell = \lceil \log(n) \rceil + p + 1$, we find $a_q(n, \ell, p) \geq q^{n-1}$ and thus a code with a single redundancy symbol exists.

**Theorem 5 (Chee et al. [4]).** For all $n, \ell, p$ and for all $q$,
\[
b_q(n, \ell, p) = q^p \cdot r_q(n - p, \ell - p).
\]

We extend this result to the LPA constraint as follows.

**Lemma 3.** For all $n, \ell, p$ and for all $q$,
\[
a_q(n, \ell, p) \leq q^{n-1} \cdot r_q(n - p + 1, \ell - p + 1).
\]

**Proof:** Via Theorem 5 and $A_q(n, \ell, p) \subseteq B_q(n, \ell, p - 1)$.

Therefore, by utilizing the bound on $k$-RLL codes in Theorem 6, we find in Corollary 4 an upper-bound on $a_q(n, \ell, p)$.

**Theorem 6 ([8]).** For all $n, k$ where $n \geq 2k$, and for all $q$,
\[
r_q(n, k) \leq q^{n-c \frac{n-2k}{q^k}}, \quad \text{for } c = \frac{\log(q-1)^2}{2q^2}.
\]

**Corollary 4.** For all $n, \ell, p$ where $n \geq 2\ell - p + 1$, and for all $q$, if there exists an $(\ell, p)$-LPA code with a single redundancy symbol, then $\ell \geq \log(n - 2\ell + p) + p - 3.5$.

Therefore, we find that Construction 2 is near the lower bound of the optimal construction. In particular, if $n \geq 2\ell - 2p + 2$, then we differ by up to 5.5 from the lower bound.

B. EXACT FORMULAS

Here, we provide interesting exact formulas for the cases of $n = \ell$ and $n \leq 2\ell - 2p + 4$. We begin with $b_q(n, n, p)$.

**Lemma 4.** For all $n, p$, and for all $q$,
\[
b_q(n, n, p) = q^n - q^p.
\]

We now address the more challenging case of $a_q(n, n, p)$.

**Theorem 7.** For all $n, p$ such that $n \geq 2p - 4$, and for all $q$,
\[
a_q(n, n, p) = q^n - \frac{q}{q-1} \cdot \sum_{d=1}^{p-1} \mu(d) \cdot q^{\left\lfloor \frac{d}{n} \right\rfloor}.
\]

where $\mu$ is the Möbius function.

This result can be extended for more cases when $n > \ell$.

**Theorem 8.** For all $n, \ell, p$ such that $n \leq 2\ell - 2p + 4$,
\[
|A_q(n, \ell, p)| = |A_q(n, \ell, p)| \cdot q^{n-\ell} \cdot (1 + (n - \ell) \cdot (1 - q^{-1})).
\]

VI. CONCLUSION

In this work, we study codes that constrain periodicity within windows of the encoded messages. We propose a construction with a single symbol of redundancy based on iteratively repairing invalid windows until a valid message is encountered. Even though the algorithm does not possess monotonic progression, we prove convergence with linear average time complexity through a reduction to an acyclic graph walk. We continue by generalizing the proposed construction to offer a trade-off between the window size and the number of additional redundancy symbols. Lastly, we study the cardinality of the constraints to both prove that the proposed construction is nearly optimal, and to mention novel exact formulas. Overall, we establish foundational constructions for constrained periodicity that may be fundamental for many different applications, such as racetrack memories.

ACKNOWLEDGMENTS

The research was Funded by the European Union. Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or European Research Council.

REFERENCES

[1] S. S. P. Parkin, M. Hayashi, and L. Thomas, “Magnetic domain-wall racetrack memory,” Science, 2008.
[2] R. Bläusing et al., “Magnetic racetrack memory: From physics to the cusp of applications within a decade,” Proc. of the IEEE, 2020.
[3] C. Zhang et al., “Hi-Fi playback: Tolerating position errors in shift operations of racetrack memory,” SIGARCH Comput. Archit. News, 2015.
[4] Y. M. Chee et al., “Coding for racetrack memories,” IEEE TIT, 2018.
[5] ——, “Reconstruction from deletions in racetrack memories,” in IEEE ITW, 2018.
[6] J. Sima and J. Bruck, “Correcting deletions in multiple-heads racetrack memories,” in IEEE ISIT, 2019.
[7] A. Kobovich, O. Leitersdorf, D. Bar-Lev, and E. Yaakobi, “Codes for constrained periodicity,” arXiv, 2022.
[8] M. Levy and E. Yaakobi, “Mutually uncorrelated codes for DNA storage,” IEEE TIT, 2019.
[9] N. J. Fine and H. S. Wilf, “Uniqueness theorems for periodic functions,” Proceedings of the American Mathematical Society, 1965.
[10] C. Choffrut and J. Karhumäki, “Combinatorics of words,” in Handbook of Formal Languages, Springer, 1997, pp. 329–438.
[11] B. H. Marcus, R. M. Roth, and P. H. Siegel, “An introduction to coding for constrained systems,” Lecture notes, 2001.