Reconstruction of Sets of Strings From Prefix/Suffix Compositions

Ryan Gabrys, *Member, IEEE*, Srilakshmi Pattabiraman, *Student Member, IEEE*, and Olgica Milenkovic, *Fellow, IEEE*

***Abstract*— The problem of reconstructing strings from substring information has found many applications due to its importance in genomic data sequencing and DNA- and polymer-based data storage. One important paradigm requires reconstructing mixtures of strings based on the union of compositions of their prefixes and suffixes, generated by mass spectrometry devices. We describe new coding methods that allow for unique joint reconstruction of subsets of strings selected from a code and provide upper and lower bounds on the asymptotic rate of the underlying codebooks. Our code constructions combine properties of binary B_h and Dyck strings that can be extended to accommodate missing substrings in the pool. As auxiliary results, we present simple entropy upper bounds for binary B_h codes and an improved bound for $h = 4$, and also describe errors that arise during mass spectrometry.**

***Index Terms*— Binary B_h codes, Dyck strings, polymer-based data storage, unique string reconstruction.**

I. Introduction

Modern digital data storage systems are facing fundamental density limits. To address the emerging needs for large-volume information archiving, it is of importance to identify new recording media that operate at the nanoscale level. Recently proposed DNA-based data storage paradigms [1], [2], [3], [4], [5], [6], [7], [8] use macromolecules as storage media and offer storage densities that are orders of magnitude higher than those of flash and optical recorders. However, these systems often come with a prohibitively high cost as well as slow and error-prone read/write platforms. To address some of these problems, several new coding solutions that aid in string assembly, dealing with asymmetries in the readout channel and reconciliation of multiple string evidence sets were introduced in [9], [10], [11], [12], [13], [14], and [15] (see also the related and follow-up lines of work [16], [17], [18], [19], [20], [21]).

As an alternative to DNA-based data storage systems, polymer-based data storage systems [1], [6] are particularly attractive due to their low synthesis cost [1]. In such platforms, two molecules of significantly different masses are synthesized to represent the bits 0 and 1, respectively. The molecules are used as building blocks in the sequential process of recording user-defined information. The obtained synthetic polymers are read by tandem mass (MS/MS) spectrometers. A mass spectrometer breaks multiple copies of the polymer uniformly at random, thereby creating prefixes and suffixes of the string of various lengths. The readout system outputs masses of these prefixes and suffixes. If the masses of all prefixes from a single string are accounted for and error-free, reconstruction is straightforward. But if multiple strings are read simultaneously and the masses of prefixes and suffixes of the same length are confusable, the problem becomes significantly more complicated. It is currently not known which combinations of coded binary strings can be distinguished from each other based on prefix-sufix masses and for which code rates it is possible to perform unique multistring reconstruction.

In a related research direction, the problem of reconstructing a string from an abstraction of its MS/MS sequencer output was considered in [22], under the name *string reconstruction from substring composition multisets*. The *composition* of a binary string is the number of 0s and the number of 1s in the string. For example, the composition of 001 equals $0^2 1^1$, indicating that 001 contains two 0s and one 1, without revealing the order of the bits. The substring composition multiset $C(\mathbf{s})$ of a string $\mathbf{s}$ is the multiset of compositions of all possible substrings of the string $\mathbf{s}$. As an illustration, the set of all substrings of 001 equals $\{0, 0, 1, 00, 01, 001\}$, and the substring composition multiset of 001 equals $\{0^1, 0^1, 1^1, 0^2, 0^1 1^1, 0^2 1^1\}$. Two modeling assumptions are used for the purpose of rigorous mathematical analysis of this reconstruction problem [22], [23], [24], [25]: a) Based on MS/MS measurements, one can uniquely infer the composition of a polymer substring from its mass; and b) When a polymer is broken down for mass spectrometry analysis, the masses of all its substrings are observed with identical frequencies.

Under the above modeling assumptions, the authors of [22] established that strings are uniquely reconstructable up to reversal, provided that the length of the strings n is ≤ 7 or one less than a prime or one less than twice a prime. The works [23], [24], [25] demonstrated that at most logarithmic code redundancy can ensure unique reconstruction of single strings drawn from codebooks based on Bertrand-Catalan strings and Reed-Solomon-like redundancy.

Manuscript received 4 October 2021; revised 20 February 2022, 15 July 2022, 29 September 2022, and 4 November 2022; accepted 7 November 2022. Date of publication 14 November 2022; date of current version 16 January 2023. The work was funded by the DARPA Molecular Informatics Program, the NSF portion of funding from the SemiSynBio program and the NSF grants CIF 2008125 and 1618366. An earlier version of this paper was presented in part at the Information Theory Workshop (ITW) 2020, Riva del Garda, Italy [DOI: 10.1109/ITW46852.2021.9457660]. The associate editor coordinating the review of this article and approving it for publication was E. Rosnes. *(Corresponding author: Olgica Milenkovic.)*

The authors are with the Department of Electrical and Computer Engineering, University of Illinois Urbana–Champaign, Urbana, IL 61801 USA (e-mail: ryan.gabrys@gmail.com; sp16@illinois.edu; milenkov@illinois.edu).

Digital Object Identifier 10.1109/TCOMM.2022.3222341

However, the assumption that MS/MS output measurements include masses of all substrings is not true in practice, as breaking the string in one rather than two locations is easier to perform. In the former case, one is presented with masses of the prefixes and suffixes. Thus, for the string 001, one would observe the multiset $\{0^1, \cancel{0^1}, 1^1, 0^2, 0^1 1^1, 0^2 1^1\}$. Furthermore, in practice, the contents of multiple strings are often read simultaneously, in which case it is not known how to associate prefixes and suffixes with their corresponding strings.

The problem addressed in this work may be formally stated as follows. Given $h \geq 2$, where $h \in \mathbb{N}$, we seek the size of the largest code $C(h)$ of binary strings of a fixed length n with a property that we refer to as *h-unique reconstructability*. In this setting, for any subcollection $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_{\bar{h}}$ of $\bar{h} \leq h$ distinct strings from $C(h)$, one is presented with the multiset union $\mathcal{M}(\mathbf{s}_1) \cup \mathcal{M}(\mathbf{s}_2) \cup \cdots \cup \mathcal{M}(\mathbf{s}_{\bar{h}})$ of the prefix-suffix composition multisets, $\mathcal{M}(\mathbf{s}_i)$, $i = 1, \ldots, \bar{h}$, of the individual strings $\mathbf{s}_i$, $i = 1, \ldots, \bar{h}$. The prefix-suffix composition multiset $\mathcal{M}(\mathbf{s})$ of a string $\mathbf{s}$ captures the weights of prefixes and suffixes of the string $\mathbf{s}$ of all lengths. Unique reconstruction refers to the property of being able to distinguish all possible $\bar{h}$-unions and nonambiguously determine the identity of the strings in the collection. Our main result provides a construction for $C(h)$ that asymptotically approaches a rate of $1/h$, under certain mild parameter constraints. The proofs of our results rely on the use of Dyck and binary B_h strings. For the latter, constructions and bounds pertaining to $h = 2$ and $h = n$ have been investigated in-depth [26], [27], [28], [29], [30], [31], [32], and we provide new nontrivial results for $h = 4$ that improve simple entropy bounds also reported in this work (see also [32], [33], [34], [35] for bounds corresponding to some special parameter choices). We also introduce some simple schemes for combating missing prefix-suffix errors in the pool.

II. Problem Statement and Preliminaries

All logarithms are taken with respect to base 2, unless stated otherwise. The symbol $[n]$ is used to denote the set $\{1, 2, \ldots, n\}$, while $[[n]]$ is used to denote the set $\{0, 1, \ldots, n\}$. A collection of i contiguous 0s in a string is, as already noted, represented by 0^i. A similar notation is used for 1s. We also find the following notation relevant to our subsequent exposition.

Let $\mathbf{s} = s_1 \ldots s_n \in \{0,1\}^n$ be a binary string of length n and let $\mathcal{M}(\mathbf{s})$ denote the composition multiset of all prefixes and suffixes of $\mathbf{s}$. For example, if $\mathbf{s} = 01101$, then, $\mathcal{M}(\mathbf{s}) = \left\{0, 01, 01^2, 0^2 1^2, 0^2 1^3, 1, 01, 01^2, 01^3, 0^2 1^3\right\}$. We denote the set of prefix and suffix compositions of $\mathbf{s}$ by $\mathcal{M}_p(\mathbf{s})$ and $\mathcal{M}_s(\mathbf{s})$, respectively. For the above string, $\mathcal{M}_p(\mathbf{s}) = \{0, 01, 01^2, 0^2 1^2, 0^2 1^3\}$ and $\mathcal{M}_s(\mathbf{s}) = \{1, 01, 01^2, 01^3, 0^2 1^3\}$.

We seek to design a binary codebook $C(n,h) \subseteq \{0,1\}^n$ so that for any collection of distinct strings $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_{\bar{h}} \in C(n,h)$ with $\bar{h} \leq h$, the multiset

$$\mathcal{M}(\mathbf{s}_1) \cup \mathcal{M}(\mathbf{s}_2) \cup \cdots \cup \mathcal{M}(\mathbf{s}_{\bar{h}}) \tag{1}$$

uniquely determines the individual strings in the collection. We refer to a code with such a property as an ***h-multicomposition code***, or an ***h-MC code***. For simplicity, we often use $\mathcal{M}(S)$ to describe the multicomposition set for $S = \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_h\}$. We also say that $C_p(n,h) \subseteq \{0,1\}^n$ is a h-prefix code if for any two distinct sets of size $\leq h$, say $S_1, S_2 \subseteq C_p$, $\mathcal{M}_p(S_1) \neq \mathcal{M}_p(S_2)$.

The next claim establishes a useful connection between our problem and the related problem of determining binary strings based on their real-valued sum.

Claim 1: Given $\mathcal{M}_p(\mathbf{s}_1) \cup \mathcal{M}_p(\mathbf{s}_2) \cup \cdots \cup \mathcal{M}_p(\mathbf{s}_h)$, one can determine $\mathbf{s}_1 + \mathbf{s}_2 + \cdots + \mathbf{s}_h \in \mathbb{R}^n$.

Proof: We prove the result for $h = 2$ as the generalization is straightforward. Suppose that $\mathbf{s}_1, \mathbf{s}_2 \in \{0,1\}^n$. Then, given $\mathcal{M}_p(\mathbf{s}_1) \cup \mathcal{M}_p(\mathbf{s}_2)$, let n_i denote the total number of ones in the two compositions of prefixes of length i in the multiset (i.e., sum of their weights). It is straightforward to see that $\mathbf{s}_1 + \mathbf{s}_2 = t_1 t_2 \ldots t_n$, where $t_i = n_i - n_{i-1}$, with $n_0 = 0$. ■

Example 2: Consider the strings $\mathbf{s}_1 = 110100$ and $\mathbf{s}_2 = 101010$, for which we have $\mathbf{s}_1 + \mathbf{s}_2 = 211110$. As before, $\mathbf{s}_1 + \mathbf{s}_2$ denotes addition over the reals, while $(\mathbf{s}_1 + \mathbf{s}_2)_i$ denotes the i^{th} symbol in the string. Clearly, $(\mathbf{s}_1 + \mathbf{s}_2)_1 = 2$, which we obtained by summing up the compositions of prefixes of length one, i.e., $1 + 1 = 2$. It is also easy to see that $(\mathbf{s}_1 + \mathbf{s}_2)_2 = (\mathbf{s}_1 + \mathbf{s}_2)_1^2 - (\mathbf{s}_1 + \mathbf{s}_2)_1$, where $(\mathbf{s}_1 + \mathbf{s}_2)_1^2$ denotes the sum of the weights of the first two symbols, or alternatively, the sum of the weights of the prefixes of length two of the strings $\mathbf{s}_1, \mathbf{s}_2$. A straightforward calculation reveals that $(\mathbf{s}_1 + \mathbf{s}_2)_2 = (2 + 1) - 2 = 1$. Other values can be determined similarly. □

The above claim provides a useful connection between our problem and the problem of designing binary B_h sequences. A ***binary*** B_h ***sequence*** is a set $\mathcal{S}_h(n)$ of binary strings of fixed length n such that for any two distinct subsets of distinct strings in $\mathcal{S}_h(n)$, say $\mathcal{S} = \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_{\bar{h}_1}\} \neq \mathcal{T} = \{\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_{\bar{h}_2}\}$, where $\bar{h}_1, \bar{h}_2 \leq h$, one has

$$\sum_{i=1}^{\bar{h}_1} \mathbf{s}_i \neq \sum_{j=1}^{\bar{h}_2} \mathbf{t}_j. \tag{2}$$

Note that although $\mathcal{S}_h(n)$ consists only of binary strings, addition is performed over the reals.

Binary B_h sequences are different from the better-known B_h and Sidon sequences (sets). A Sidon (B_h) sequence is a single sequence of integers such that the sums of two elements (h elements) of the sequence are all distinct [36]. Addition can be performed over the integers or over finite fields [37], [38]. To avoid possible confusion with the naming convention, we henceforth refer to the set $\mathcal{S}_h(n) \subseteq \mathbb{R}^n$ as a ***binary*** B_h ***code*** of length n. For shorthand, we also refer to any codestring $\mathbf{s} \in \mathcal{S}_h(n)$ as a binary B_h string.

If S is a B_h sequence over $\mathbb{F}_2^d$, for some positive integer d, then it is also a binary B_h code. However, the opposite is not necessarily true; hence, arguments typically used to derive upper bounds for B_h sequences do not carry over to binary B_h codes.

Example 3: Consider the set $\mathcal{S}_2(6) = \{110100, 101010, 110010\}$. It is easy to verify that the sums of pairs of strings in $\mathcal{S}_2(6) \subset \mathbb{R}^6$ are distinct. Thus, $\mathcal{S}_2(6)$ is a binary B_2 code.

However, $\mathcal{S}'_2(6) = \{110100, 101010, 110010, 101100\}$ is not a binary B_2 code since $110100 + 101010 = 110010 + 101100 = 211110$. $\square$

Based on Claim 1, it is easy to identify two sufficient conditions for a collection of binary strings to be an h-MC code, where $\bar{h} \le h$.

1) **Condition 1:** One can recover $\mathcal{M}_p(\mathbf{s}_1) \cup \dots \cup \mathcal{M}_p(\mathbf{s}_{\bar{h}})$ from $\mathcal{M}(\mathbf{s}_1) \cup \dots \cup \mathcal{M}(\mathbf{s}_{\bar{h}})$, for any choice of distinct codestrings $\mathbf{s}_1, \dots, \mathbf{s}_{\bar{h}}$; and
2) **Condition 2:** The codestrings $\mathbf{s}_1, \dots, \mathbf{s}_{\bar{h}}$ belong to a binary B_h code $\mathcal{S}_h(n)$.

These observations will be used to construct h-MC codes in Section III. The condition that the codestrings in an MC code belong to a B_h-code is not necessary. For example, consider the strings $\mathbf{s}_1 = 011$, $\mathbf{s}_2 = 000$, $\mathbf{s}_3 = 001$, $\mathbf{s}_4 = 010$. Then, $\mathbf{s}_1 + \mathbf{s}_2 = 011 = \mathbf{s}_3 + \mathbf{s}_4$, but $01^2 \in \mathcal{M}(\mathbf{s}_1) \cup \mathcal{M}(\mathbf{s}_2)$ and $01^2 \notin \mathcal{M}(\mathbf{s}_3) \cup \mathcal{M}(\mathbf{s}_4)$, so that $\{\mathbf{s}_1, \mathbf{s}_2\}$ and $\{\mathbf{s}_3, \mathbf{s}_4\}$ are not confusable.

We show next that for sufficiently large code lengths, the maximum rate of an h-MC code is at least $\frac{1}{h}$. In comparison, the best currently known upper bound on the rate of binary B_2 codes is .5753 [39]. For related bounds and bounds for slightly differently defined binary B_h codes, the reader is referred to [31], [33], and [40].

III. A Constructive Lower Bound for h-MC Codes

We start with a binary B_h code and introduce redundancy into the underlying strings to ensure that given the multicomposition set of at most h strings, one can separate the prefixes from the suffixes. Then, given the set of prefixes, one can use the same idea behind Claim 1 to recover the sum of the h codestrings and hence the codestrings themselves.

Let $\mathcal{S}_h(n) \subseteq \mathbb{R}^n$ be a binary B_h code. One way to construct the code $\mathcal{S}_h(n)$ is to use the columns of a parity-check matrix of a linear code with minimum Hamming distance $\ge 2h+1$ [41]. This follows from the simple observation that no two distinct collections of h distinct columns of a binary parity-check matrix of a code with $d_{\min} \ge 2h+1$ can have the same sum modulo 2, and consequently, cannot have the same real-valued sum either. For such binary codes $\mathcal{S}_h(n)$ of largest size, one can show that the asymptotic code rate satisfies $\lim_{n\to\infty} \frac{1}{n} \log |\mathcal{S}_h(n)| = \frac{1}{h}$.

For our problem and the underlying approach for solving it, we also have to make use of Dyck strings.

Definition 4: A string $\mathbf{s} \in \mathbb{R}^N$ of even length N is a ***Dyck string*** if its weight satisfies $\text{wt}(\mathbf{s}) = \frac{N}{2}$, and for $i \in [N-1]$, $\text{wt}(s_1 s_2 \dots s_i) \ge \left\lceil \frac{i}{2} \right\rceil$.

The approach for generating the binary code $C(N,h) \subseteq \mathbb{R}^N$ is to ensure that: 1) A string $\mathbf{s} \in C(N,h)$ is a Dyck string; 2) the set $C(N,h)$ is a binary B_h code of length N. The first property guarantees that the mixtures of prefixes and suffixes can be partitioned into two sets, one containing all the prefixes and another containing all the suffixes. The second property ensures that given the prefix set (or, alternatively, the suffix set) one can recover the codestrings using the simple observation that the prefixes uniquely determine the real-valued sum of the strings in the mixture. We illustrate these observations with an example.

Example 5: Consider the binary B_2 code $\mathcal{S}_2(6) = \{110100, 101010, 110010\}$. Clearly, all three strings are Dyck strings as their prefixes of any length contain at least as many ones as zeros.

Next, write $\mathbf{s}_1 = 110100$ and $\mathbf{s}_2 = 101010$, so that $\mathcal{M}(\mathbf{s}_1) \cup \mathcal{M}(\mathbf{s}_2) = \{1, 1, 01,\ 1^2, 01^2,\ 01^2, 0^21^2,\ 01^3, 0^21^3, 0^21^3,\ 0^31^3, 0^31^3,\ 0^31^3, 0^31^3,\ 0^31^2, 0^31^2, 0^21^2, 0^31, 0^21, 0^21, 01, 0^2, 0, 0\}$. Since $\mathbf{s}_1$ and $\mathbf{s}_2$ are Dyck strings, each of the string prefixes must have at least as many 1s as 0s. Similarly, each suffix must have at least as many 0s as 1s. It follows from this observation that one can easily recover the multiset $\mathcal{M}_p(\mathbf{s}_1) \cup \mathcal{M}_p(\mathbf{s}_2) = \{1, 1, 01,\ 1^2, 01^2,\ 01^2, 0^21^2,\ 01^3, 0^21^3, 0^21^3,\ 0^31^3, 0^31^3\}$. Claim 1 ensures that given $\mathcal{M}_p(\mathbf{s}_1) \cup \mathcal{M}_p(\mathbf{s}_2)$, one can determine $\mathbf{s}_1 + \mathbf{s}_2 = 211110$. Since $\mathcal{S}_2(6)$ is a binary B_2 code, the sum $\mathbf{s}_1 + \mathbf{s}_2$ uniquely determines the strings $\mathbf{s}_1$ and $\mathbf{s}_2$. $\square$

The next claim establishes the formal result that if the code $C(N,h)$ satisfies these two properties, then it is an h-MC code.

Claim 6: Suppose that $C(N,h)$ is a binary B_h code where for any $\mathbf{s} \in C(N,h)$, the defining Dyck property holds. Then, $C(N,h)$ is an h-MC code.

Proof: Similar to Claim 1, we prove the statement for $h = 2$, since the extension for general h is straightforward. In light of Claim 1, we need to show that the Dyck property allows us to uniquely recover $\mathcal{M}_p(\mathbf{s}_1) \cup \mathcal{M}_p(\mathbf{s}_2)$ from $\mathcal{M}(\mathbf{s}_1) \cup \mathcal{M}(\mathbf{s}_2)$. To see that this is indeed possible, observe that based on the Dyck property both prefixes of length i in $\mathcal{M}(\mathbf{s}_1) \cup \mathcal{M}(\mathbf{s}_2)$ have at least $\lceil \frac{i}{2} \rceil$ 1s whereas both suffixes of length i in $\mathcal{M}(\mathbf{s}_1) \cup \mathcal{M}(\mathbf{s}_2)$ have at most $\lfloor \frac{i}{2} \rfloor$ 1s. $\blacksquare$

To maximize the rate of the coding scheme and combine the two constraints that h-**MC** strings need to satisfy, we use two ideas. First, we use B_h binary strings obtained from appropriate parity-check matrices of binary error-correcting codes with minimum distance $\ge 2h+1$, parsed into blocks (substrings) that allow us to tightly control the number of ones (weights) of the codestrings via balancing. Upon balancing blocks in each codestring $\mathbf{s} \in \mathcal{S}_h(n)$ we append $\mathcal{O}(\sqrt{n})$ bits of redundancy both to the beginning and to the end of $\mathbf{s}$ so that the resulting string has length $N = n + \mathcal{O}(\sqrt{n})$. Second, rather than work directly with the weights of strings we use the running digital sums (RDSs). For a binary string $\mathbf{s}$, the RDS up to coordinate i is defined as $R(\mathbf{s})_i = 2\text{wt}(s_1 s_2 \dots s_i) - i$. If the subscript i is omitted, then $R(\mathbf{s}) = 2\text{wt}(\mathbf{s}) - |\mathbf{s}|$, where $|\mathbf{s}|$ denotes the length of $\mathbf{s}$. Furthermore, using the running digital sum, the Dyck constraint can be rewritten as $\text{wt}(\mathbf{s}) = \lceil \frac{n}{2} \rceil$ and $R(\mathbf{s})_i \ge 0$, $i \in [n]$. It follows that for any $\epsilon > 0$ and n sufficiently large, we have $\frac{1}{N} \log |C(N,h)| = \frac{1}{n + \kappa\sqrt{n}} \log |\mathcal{S}_h(n)| = \frac{1}{h} - \epsilon$, where κ is a constant.

The balancing procedure operates as follows: Let $\mathbf{s} \in \mathcal{S}_h(n)$ and for simplicity assume that $\sqrt{n}$ is an even integer. Start by parsing $\mathbf{s}$ into blocks $\mathbf{s}_i$ of length $\sqrt{n}$, $i = 1, \dots, \sqrt{n}$, so that $\mathbf{s} = \mathbf{s}_1 \mathbf{s}_2 \dots \mathbf{s}_{\sqrt{n}}$. Using $\mathbf{s}$ construct an auxiliary string $\mathbf{u} = \mathbf{u}_1 \mathbf{u}_2 \dots \mathbf{u}_{\sqrt{n}}$ that is “approximately” balanced following an idea similar to Knuth's balancing [42], which operates on blocks rather than individual symbols. To this end, initialize $\mathbf{u}_1 = \mathbf{s}_1$; for binary strings $\mathbf{u}$, we use $\overline{\mathbf{u}}$ to denote the binary complement of $\mathbf{u}$. For $j \in \{2, 3, \ldots, \sqrt{n}\}$, let

$$\mathbf{u}_j = \begin{cases} \mathbf{s}_j, & \text{if } R(\mathbf{u}_1 \ldots \mathbf{u}_{j-1}) < 0, \text{ and } R(\mathbf{s}_j) \geq 0, \\ \overline{\mathbf{s}}_j, & \text{if } R(\mathbf{u}_1 \ldots \mathbf{u}_{j-1}) < 0, \text{ and } R(\mathbf{s}_j) < 0, \\ \mathbf{s}_j, & \text{if } R(\mathbf{u}_1 \ldots \mathbf{u}_{j-1}) \geq 0, \text{ and } R(\mathbf{s}_j) < 0, \\ \overline{\mathbf{s}}_j, & \text{if } R(\mathbf{u}_1 \ldots \mathbf{u}_{j-1}) \geq 0, \text{ and } R(\mathbf{s}_j) \geq 0. \end{cases} \tag{3}$$

Claim 7: For any $j \in [\sqrt{n}]$, $|R(\mathbf{u}_1 \ldots \mathbf{u}_j)| \leq \sqrt{n}$.

Lemma 8: For any $i \in [n]$, $|R(\mathbf{u})_i| \leq \frac{3}{2}\sqrt{n}$. Hence, the RDS of any prefix of $\mathbf{u}$ does not exceed $\frac{3}{2}\sqrt{n}$ in absolute value.

Proof: Suppose to the contrary that $|R(\mathbf{u})_i| > \frac{3\sqrt{n}}{2}$. For simplicity, we will only consider the case $R(\mathbf{u})_i > \frac{3\sqrt{n}}{2}$, as the other case can be handled similarly. Next, assume that $j \in [n]$ is the smallest index for which $R(\mathbf{u})_j > \frac{3\sqrt{n}}{2}$ and that $R(\mathbf{u})_j = \frac{3\sqrt{n}}{2} + 1$. Now, let $j = k_1\sqrt{n} + k_2$, where $0 \leq k_2 < \sqrt{n}$. According to Claim 7, since $R(\mathbf{u})_j = \frac{3\sqrt{n}}{2} + 1$ and $k_2 < \sqrt{n}$, we have $\frac{\sqrt{n}}{2} < R(\mathbf{u}_1 \ldots \mathbf{u}_{k_1-1}) \leq \sqrt{n}$. Based on (3), and since $R(\mathbf{u}_1 \ldots \mathbf{u}_{k_1-1}) > \frac{\sqrt{n}}{2}$, it follows that $R(\mathbf{u}_{k_1}) \leq 0$ so that $-\sqrt{n} \leq R(\mathbf{u}_{k_1})_\ell \leq \frac{\sqrt{n}}{2}$, for any $\ell \in [\sqrt{n}]$. Combining the two inequalities, we arrive at $R(\mathbf{u})_{k_1\sqrt{n}+k_2} \leq \frac{3\sqrt{n}}{2}$, a contradiction. ■

We now describe our encoder. Let $\mathbf{u} \in \{0,1\}^n$ be the string which is the result of the procedure described in (3), and suppose that $\mathbf{r} \in \{0,1\}^{\sqrt{n}}$ is such that for any $j \in [\sqrt{n}]$,

$$r_j = \begin{cases} 1, & \text{if } \mathbf{u}_j \neq \mathbf{s}_j, \\ 0, & \text{if } \mathbf{u}_j = \mathbf{s}_j. \end{cases} \tag{4}$$

Using $\mathbf{r}$, we now form a string $\mathbf{s} \in C(N,h)$, where $N = n + \frac{17}{2}\sqrt{n}$, and assume for simplicity that N is an even integer. The following claim is used in our subsequent analysis.

Claim 9: Let $\mathbf{v} = \mathbf{1}^{\frac{5}{2}\sqrt{n}}\mathbf{r}\mathbf{u} \in \{0,1\}^{n+\frac{7}{2}\sqrt{n}}$. Then, for any $i \in [n + \frac{7}{2}\sqrt{n}]$, $|R(\mathbf{v})_i| \leq 5\sqrt{n}$. Furthermore, for any $i \in [n + \frac{7}{2}\sqrt{n}]$, $R(\mathbf{v})_i > 0$.

Next, we append redundant bits to the string $\mathbf{v}$ described in Claim 9 in order to get a binary string $\mathbf{s}$ of length N which is a Dyck string.[1] This results in the following claim.

Claim 10: Let $N = n + \frac{17}{2}\sqrt{n}$ be an even integer and let $\mathbf{v} = \mathbf{1}^{\frac{5}{2}\sqrt{n}}\mathbf{r}\mathbf{u} \in \{0,1\}^{n+\frac{7}{2}\sqrt{n}}$ be as in Claim 9. Suppose that $w = \text{wt}(\mathbf{v})$. Then, the string $\mathbf{s} = \mathbf{v}\mathbf{1}^{\frac{N}{2}-w}\mathbf{0}^{\frac{N}{2}-(|\mathbf{v}|-w)}$ is a Dyck string.

Now, assume that the binary code $C(N,h) \subseteq \mathbb{R}^N$ is constructed according to the procedure outlined in Claim 10 and once again assume that $N = n + \frac{17}{2}\sqrt{n}$ is an even integer. The next theorem establishes the correctness of our construction.

Theorem 11: Suppose that $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_h \in C(N,h)$, where $C(N,h)$ is constructed according to the balancing procedure operating on some selected codebook of binary B_h strings. Then, given $\mathcal{M}(\mathbf{s}_1) \cup \mathcal{M}(\mathbf{s}_2) \cup \cdots \cup \mathcal{M}(\mathbf{s}_h)$, we can uniquely determine $\{\mathbf{s}_1, \ldots, \mathbf{s}_h\}$. Furthermore, for any $\epsilon > 0$, there exists a $n_\epsilon > 0$ such that for all $N \geq n_\epsilon$, $\frac{1}{N}\log|C(N,h)| \geq \frac{1}{h} - \epsilon$.

Proof: We prove the result for $h = 2$, as the extension for general values of h is straightforward. According to Claims 6 and 10, we can recover $\mathcal{M}_p(\mathbf{s}_1) \cup \mathcal{M}_p(\mathbf{s}_2)$ from $\mathcal{M}(\mathbf{s}_1) \cup \mathcal{M}(\mathbf{s}_2)$ since $\mathbf{s}_1, \mathbf{s}_2$ are Dyck strings. From $\mathcal{M}_p(\mathbf{s}_1) \cup \mathcal{M}_p(\mathbf{s}_2)$, we can recover $\mathbf{s}_1 + \mathbf{s}_2$ according to Claim 1. Given $\mathbf{s}_1 = \mathbf{1}^{\frac{5}{2}\sqrt{n}}\mathbf{r}_1\mathbf{u}_1\mathbf{1}^{\frac{N}{2}-w_1}\mathbf{0}^{\frac{N}{2}-(|\mathbf{v}_1|-w_1)}$, $\mathbf{s}_2 = \mathbf{1}^{\frac{5}{2}\sqrt{n}}\mathbf{r}_2\mathbf{u}_2\mathbf{1}^{\frac{N}{2}-w_2}\mathbf{0}^{\frac{N}{2}-(|\mathbf{v}_2|-w_2)}$, from the first $n + \frac{7}{2}\sqrt{n}$ coordinates of $\mathbf{s}_1 + \mathbf{s}_2$ we can recover $(\mathbf{r}_1 + \mathbf{r}_2, \mathbf{u}_1 + \mathbf{u}_2) \bmod 2$. Note that here we only make use of the parity information although we are presented with real-valued sums, as we wish to recover the binary indicator $\mathbf{r}$ string for block complementation and the binary blocks themselves.

For simplicity, and with a slight abuse of notation, we write $\mathbf{u} = \mathbf{u}_1 + \mathbf{u}_2 \bmod 2 = \mathbf{u}_1\mathbf{u}_2 \ldots \mathbf{u}_{\sqrt{n}}$ and $\mathbf{r} = \mathbf{r}_1 + \mathbf{r}_2 \bmod 2 = \mathbf{r}_1 \ldots \mathbf{r}_{\sqrt{n}}$. Let $\tilde{\mathbf{u}} = \tilde{\mathbf{u}}_1 \ldots \tilde{\mathbf{u}}_{\sqrt{n}}$. Then, for $j \in [\sqrt{n}]$,

$$\tilde{\mathbf{u}}_j = \begin{cases} \mathbf{u}_j, & \text{if } r_j = 0, \\ \overline{\mathbf{u}}_j & \text{if } r_j = 1. \end{cases}$$

It is straightforward to verify from (3) that $\tilde{\mathbf{u}} = \mathbf{s}_1 + \mathbf{s}_2 \bmod 2$. Since $\mathbf{s}_1, \mathbf{s}_2 \in \mathcal{S}_2(n)$ are codestrings of a binary B_h codebook comprising columns of a binary error-correcting code of appropriate parameters, we can recover $\mathbf{s}_1$ and $\mathbf{s}_2$ from $\tilde{\mathbf{u}}$. This concludes the proof. ■

In what follows, to improve our understanding of the maximum asymptotic rate of h-MC codes, we describe straightforward general entropy bounds and derive a (tighter) upper bound for binary B_4 codes that outperforms the entropy bound. These results imply upper bounds on the rates of h-MC codes which are not necessarily constructed using Theorem 11, centered around binary error-correcting codes and general constructions of binary B_h codes. For example, it is known [39] that the maximum rate of a binary B_2 sequence is at most 0.5753 which implies that the maximum rate of any 2-MC code using binary B_2 codes is at most 0.5753. The interested reader is referred to a selected collection of entropy and other bounds for related notions of binary B_h strings in [31], [33], and [40].

A. New Upper Bounds on Binary B_4 Sequences (Binary B_4 Codes)

We extend and generalize the idea used in [43] and [28] to obtain an upper bound on the maximum rate of a binary B_h code, and $h = 4$ in particular. We first introduce the relevant notation before describing our main result in Theorem 14.

Let $B_h(n)$ denote a binary B_h code of length n. Since $B_h(n)$ is a binary B_h code it follows that for any two distinct sets of codestrings, say $\{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_h\}, \{\mathbf{s}'_1, \mathbf{s}'_2, \ldots, \mathbf{s}'_h\} \subseteq B_h(n)$, we have $\sum_{j=1}^h \mathbf{s}_j \neq \sum_{j=1}^h \mathbf{s}'_j$, where addition is over the reals.

Let $H(h)$ denote the entropy of the Binomial distribution with parameters $(h, \frac{1}{2})$,

$$H(h) = -\sum_{k=0}^{h} \binom{h}{k} \left(\frac{1}{2}\right)^h \log_2\left(\binom{h}{k}\left(\frac{1}{2}\right)^h\right). \tag{5}$$

[1]Splitting the string into blocks of length m and then performing the approximate balancing task over these blocks would incur a redundancy of $\frac{n}{m} + cm$, where c is a constant. The redundancy is minimized when the summands are of the same order, $\sqrt{n}$, which justifies the choice for the length of the parts.

It is well-known [43] that the above entropy can be written as

$$H(h) = \frac{1}{2}\log_2\left(2\pi e\frac{h}{4}\right) + \mathcal{O}\left(\frac{1}{h}\right),$$

and that the entropy of a Binomial distribution with parameters (h, p), $0 < p \leq \frac{1}{2}$ is maximized for $p = \frac{1}{2}$. A straightforward upper bound for general h follows directly from the ideas in [43] and [28], which assert that one can impose a uniform (probabilistic) model on the product set of codestrings in B_2 and then employ the entropy of each coordinate in the 2-sum. In our case, we use the entropy of a Binomial random variable with parameters $(h, \frac{1}{2})$ to bound the contribution of each coordinate. This approach results in an asymptotic upper bound on the rate of the form $\frac{1}{h}H(h)$, where $H(h)$ is the entropy of a Binomial random variable as defined in the previous equation.

As will be discussed in more details, for $h = 4$, rather than work directly with the prefixes of codestrings as suggested in [28] for the case $h = 2$, we instead use the sum of prefixes of codestrings in a binary B_4 code. Then, through simple counting argument, we arrive at Theorem 14. We outline the argument for general even-valued h and specialize the bound to $h = 4$ only in the last step, since the proposed approach can potentially lead to improved bounds for larger h through tighter bounds on the size of specific sets used in the proof.

Let $\mathcal{A}_{h/2}$ denote the set of sums of any collection of $h/2$ prefixes of distinct codestrings from $B_h(n)$, where $n = a + b$ and a and b are the lengths of the prefixes and suffixes, respectively:

$$\mathcal{A}_{h/2} = \{\mathbf{a}_1 + \cdots + \mathbf{a}_{h/2} \in \{0, 1, \ldots, h/2\}^a : \\ (\mathbf{a}_1\mathbf{b}_1), \ldots, (\mathbf{a}_{h/2}\mathbf{b}_{h/2}) \in B_h(n)\}.$$

For $\boldsymbol{c} \in \mathcal{A}_{h/2}$, define $\mathcal{B}_{\boldsymbol{c}}$ as

$$\mathcal{B}_{\boldsymbol{c}} = \{\mathbf{b}_1 + \cdots + \mathbf{b}_{h/2} \in \{0, 1, \ldots, h/2\}^b : \\ (\mathbf{a}_1\mathbf{b}_1), \ldots, (\mathbf{a}_{h/2}\mathbf{b}_{h/2}) \in B_h(n), \mathbf{a}_1 + \cdots + \mathbf{a}_{h/2} = \boldsymbol{c}\},$$

and recall that, by definition, $B_h(n)$ has the property that any collection of $\leq h$ distinct codestrings in $B_h(n)$ has a distinct sum. This implies that no two strings in $\mathcal{B}_{\boldsymbol{c}}$ are the same.

We proceed by proving the next claim.

Claim 12: Let $\boldsymbol{c}_1, \boldsymbol{c}_2 \in \mathcal{A}_{h/2}$. Then for any $\boldsymbol{d}_1, \boldsymbol{d}_2 \in \mathcal{B}_{\boldsymbol{c}_1}$ and $\boldsymbol{d}_3, \boldsymbol{d}_4 \in \mathcal{B}_{\boldsymbol{c}_2}$,

$$\boldsymbol{d}_1 - \boldsymbol{d}_2 \neq \boldsymbol{d}_3 - \boldsymbol{d}_4, \tag{6}$$

where subtraction is performed over the reals.

Proof: Suppose, to the contrary, that (6) is an equality. Then $(\boldsymbol{c}_1\boldsymbol{d}_1) + (\boldsymbol{c}_2\boldsymbol{d}_4) = (\boldsymbol{c}_2\boldsymbol{d}_3) + (\boldsymbol{c}_1\boldsymbol{d}_2)$. In this case, we may write

$$\mathbf{a}_1^{(1)} + \mathbf{a}_2^{(1)} + \cdots + \mathbf{a}_{h/2}^{(1)} = \boldsymbol{c}_1,\ \mathbf{a}_1^{(2)} + \mathbf{a}_2^{(2)} + \cdots + \mathbf{a}_{h/2}^{(2)} = \boldsymbol{c}_2,$$
$$\mathbf{b}_1^{(1)} + \mathbf{b}_2^{(1)} + \cdots + \mathbf{b}_{h/2}^{(1)} = \mathbf{d}_1,\ \mathbf{b}_1^{(2)} + \mathbf{b}_2^{(2)} + \cdots + \mathbf{b}_{h/2}^{(2)} = \boldsymbol{d}_2,$$
$$\mathbf{b}_1^{(3)} + \mathbf{b}_2^{(3)} + \cdots + \mathbf{b}_{h/2}^{(3)} = \boldsymbol{d}_3,\ \ \mathbf{b}_1^{(4)} + \mathbf{b}_2^{(4)} + \cdots + \mathbf{b}_{h/2}^{(4)} = \boldsymbol{d}_4.$$

If the assumption above holds, it follows that

$$\begin{aligned}&((\mathbf{a}_1^{(1)}\mathbf{b}_1^{(1)}) + \cdots + (\mathbf{a}_{h/2}^{(1)}\mathbf{b}_{h/2}^{(1)})) \\ &\quad + ((\mathbf{a}_1^{(2)}\mathbf{b}_1^{(4)}) + \cdots + (\mathbf{a}_{h/2}^{(2)}\mathbf{b}_{h/2}^{(4)})) \\ &= ((\mathbf{a}_1^{(2)}\mathbf{b}_1^{(3)}) + \cdots + (\mathbf{a}_{h/2}^{(2)}\mathbf{b}_{h/2}^{(3)})) \\ &\quad + ((\mathbf{a}_1^{(1)}\mathbf{b}_1^{(2)}) + \cdots + (\mathbf{a}_{h/2}^{(1)}\mathbf{b}_{h/2}^{(2)})),\end{aligned}$$

where $(\mathbf{a}_1^{(1)}\mathbf{b}_1^{(1)}), \ldots, (\mathbf{a}_{h/2}^{(1)}\mathbf{b}_{h/2}^{(1)}), \ldots, (\mathbf{a}_{h/2}^{(1)}\mathbf{b}_{h/2}^{(2)}) \in B_h(n)$, a contradiction. ■

Based on the result of the previous claim, we focus on the differences between elements in the set $\mathcal{B}_{\boldsymbol{c}}$ and define the multiset $\mathcal{D} = \{\boldsymbol{d}_1 - \boldsymbol{d}_2 \in \{-h/2, \ldots, 0, \ldots, h/2\}^b : \exists \boldsymbol{c} \text{ s.t. } \boldsymbol{d}_1, \boldsymbol{d}_2 \in \mathcal{B}_{\boldsymbol{c}}\}$.

Claim 13: Any nonzero $\mathbf{d} \in \{-h/2, \ldots, 0, \ldots h/2\}^b$ appears at most once in $\mathcal{D}$, while the all-zero vector appears $\binom{|B_h(n)|}{h/2}$ times.

Proof: The first statement follows immediately from Claim 12. For the second claim, since $B_h(n)$ is a binary B_h code, it follows that the sum of any distinct $h/2$ codestrings from $B_h(n)$ is necessarily unique and hence $\boldsymbol{d}_1 - \boldsymbol{d}_2 = \mathbf{0}$ if and only if $\boldsymbol{d}_1 = \boldsymbol{d}_2$. Since $\sum_{\boldsymbol{c} \in \mathcal{A}_{h/2}} |\mathcal{B}_{\boldsymbol{c}}| = \binom{|B_h(n)|}{h/2}$, the result follows. ■

Let $\mathcal{D}_s = \{\mathbf{d} : \mathbf{d} \in \mathcal{D}\}$ be the *set* containing the elements in $\mathcal{D}$ except for the all-zero strings (which are removed) and recall that b stands for the length of the strings in $\mathcal{D}$ and $\mathcal{D}_s$. More formally, let $\mathcal{D}_s$ stand for

$$\{(\boldsymbol{x}_1 + \cdots + \boldsymbol{x}_{h/2}) - \left(\boldsymbol{x}_1' + \cdots + \boldsymbol{x}_{h/2}'\right) \neq 0 : \\ \exists\, \boldsymbol{c} \text{ s.t. } \boldsymbol{x}_1 + \ldots + \boldsymbol{x}_{h/2},\ \boldsymbol{x}_1' + \ldots + \boldsymbol{x}_{h/2}' \in \mathcal{B}_{\mathbf{c}}\},$$

where addition is performed over the reals.

It is straightforward to see that $|\mathcal{D}_s| \leq (h+1)^b$. This bound suffices to obtain an upper bound on the size of B_h codes for $h = 4$ that outperforms the entropy bound described at the beginning of this section. For even values of $h \geq 6$, a tighter bound on $|\mathcal{D}_s|$ is needed, akin to the one derived in [43] and [28] that relies on estimating the probabilities of zero and nonzero symbols in the individual coordinates of the sum of codestrings.

Theorem 14: For $h = 4$, the maximum asymptotic rate of a binary B_4 code is bounded from above by 0.471.

Proof: Once again, we outline the argument for general even h and specialize the proof for $h = 4$ at the last step when invoking the simple bound for $|\mathcal{D}_s|$. By definition,

$$\binom{|B_h(n)|}{h/2} = |\mathcal{B}_{\boldsymbol{c}_1}| + |\mathcal{B}_{\boldsymbol{c}_2}| + \cdots + |\mathcal{B}_{\boldsymbol{c}_{|\mathcal{A}_{h/2}|}}|.$$

From the previous equation and the convexity of the function x^2, we have

$$|\mathcal{D}| = \sum_{j=1}^{|\mathcal{A}_{h/2}|} |\mathcal{B}_{\boldsymbol{c}_j}|^2 \geq \frac{\binom{|B_h(n)|}{h/2}^2}{|\mathcal{A}_{h/2}|} \approx \frac{|B_h(n)|^h}{|\mathcal{A}_{h/2}|},$$

where we ignored constants involving h. Since according to Claim 13 the all-zero vector appears at most $|B_h(n)|^{h/2}$ times and all the other vectors appear at most once in $\mathcal{D}$,

$$|B_h(a+b)|^h \lessapprox |\mathcal{D}|\,|\mathcal{A}_{h/2}| = |\mathcal{D}\setminus\mathcal{D}_s||\mathcal{A}_{h/2}| + |\mathcal{D}_s||\mathcal{A}_{h/2}| \leq |B_h(a+b)|^{h/2} 2^{a\log_2(\frac{h}{2}+1)} + 2^{b\log_2(h+1)+a\log_2(\frac{h}{2}+1)},$$

where we used the facts that $|\mathcal{A}_{h/2}| \leq \left(\frac{h}{2}+1\right)^a$ and $|\mathcal{D}_s| \leq (h+1)^b$. Let $b = a\frac{\log_2(\frac{h}{2}+1)}{\log_2(h+1)}$. Then, the previous expression can be rewritten as:

$$|B_h(a+b)|^h \lessapprox |B_h(a+b)|^{h/2}\, 2^{a\log_2(\frac{h}{2}+1)} + 2^{2a\log_2(\frac{h}{2}+1)}.$$

It then follows that $|B_h(a+b)|^{h/2} \lessapprox 2^{a\log_2(\frac{h}{2}+1)}$. To see why this bound holds, note that either (i) $\max\left\{2^{2a\log_2(\frac{h}{2}+1)}, |B_h(a+b)|^{h/2}\, 2^{a\log_2(\frac{h}{2}+1)}\right\} = 2^{2a\log_2(\frac{h}{2}+1)}$ or (ii) $\max\left\{2^{2a\log_2(\frac{h}{2}+1)}, |B_h(a+b)|^{h/2}\, 2^{a\log_2(\frac{h}{2}+1)}\right\} = |B_h(a+b)|^{h/2}\, 2^{a\log_2(\frac{h}{2}+1)}$. Therefore, by ignoring constants, we can write $\frac{\log|B_h(a+b)|}{a+b} \lessapprox \frac{2}{h}\frac{a\log_2(\frac{h}{2}+1)}{a+b}$. Consequently, the asymptotic rate of a binary B_h codes is at most $\frac{2/h\log_2(\frac{h}{2}+1)}{1+\frac{\log_2(\frac{h}{2}+1)}{\log_2(h+1)}}$. For $h=4$, this bound reduces to $\frac{1/2\log_2(3)}{1+\frac{\log_2(3)}{\log_2(5)}} = .471$. ■

For $h=4$, the entropy bound $\frac{1}{h}H(h)$ equals $.5077$, whereas the bound derived above gives the value $.471$.

IV. Upper Bounds on h-MC Codes

Next, we derive an upper bound on the maximum rate of an h-MC code. To this end, recall that $C_p \subseteq \{0,1\}^n$ is a h-prefix code if for any two distinct string subsets of sizes $\bar{h} \leq h$, say $\mathcal{S}_1, \mathcal{S}_2 \subseteq C_p$, $\mathcal{M}_p(\mathcal{S}_1) \neq \mathcal{M}_p(\mathcal{S}_2)$. Let $C_h^{(MC)}(n)$ be the size of the largest h-MC code of codelength n and suppose that $C_h^{(p)}(n)$ is the size of the largest h-prefix code of codelength n. Formally, we use $R_h^{(MC)}$ to denote the maximum asymptotic rate of an h-MC code,

$$R_h^{(MC)} = \lim_{n\to\infty}\sup\frac{1}{n}\log|C_h^{(MC)}(n)|.$$

We show next that when h is an even constant, $R_h^{(MC)} \leq 1 - \frac{1}{2}\left(\frac{1}{1+\frac{1}{h}}\right)$. Once again, for simplicity of exposition, we focus on the case $h=2$ before considering the general result.

The next lemma states that in order to derive an upper bound on the quantity $R_h^{(MC)}$, we can limit our attention to prefix codes.

Lemma 15: For any $\epsilon > 0$, there exists an $n_\epsilon \geq 1$ such that for all $n \geq n_\epsilon$, one has

$$\frac{1}{n}\log|C_h^{(MC)}(n)| \leq \frac{1}{n}\log|C_h^{(p)}(n)| + \epsilon.$$

Proof: To simplify the discussion, we focus on the case $h=2$; the extension to $h>2$ is straightforward. For $w \in [n]$, let $C_2^{(w)}(n) \subseteq C_h^{(MC)}(n)$ denote the set of codestrings of weight w in $C_2^{(MC)}(n)$. By the pigeon-hole principle, there exists a $w^* \in [n]$ such that $|C_2^{(w^*)}(n)| \geq \frac{1}{n}|C_2^{(MC)}(n)|$. Given two codestrings in $C_2^{(w^*)}(n)$, say $\mathcal{S} = \{\mathbf{s}_1, \mathbf{s}_2\}$, we can easily determine $\mathcal{M}_p(\mathcal{S})$. Assuming that only the prefix composition set is known, the set $\mathcal{M}_s(\mathcal{S})$ can be derived as follows. To determine the compositions of suffixes of length i, for $i \in [n]$, we subtract from w^* the number of ones in each prefix of length $n-i$. For instance, suppose the compositions of prefixes of length $n-1$ of $\mathbf{s}_1, \mathbf{s}_2$ are $\left\{\{1^{w^*}, 0^{n-w^*-1}\}, \{1^{w^*-1}, 0^{n-w^*}\}\right\}$. Then, the length-1 suffixes of $\mathbf{s}_1, \mathbf{s}_2$ are $\left\{\{1\},\{0\}\right\}$. This implies that $n|C_2^{(p)}(n)| \geq n|C_2^{(w^*)}(n)| \geq |C_2^{(MC)}(n)|$, which establishes the desired result. ■

Let us first examine the case $h=2$. For any $\mathbf{s} \in C_2^{(p)}(n)$, we write $\mathbf{s}$ as $\mathbf{s} = \mathbf{ab} \in C_2^{(p)}(n)$, where $\mathbf{a} \in \{0,1\}^{\alpha n}$ equals the prefix of αn symbols of $\mathbf{s}$ while $\mathbf{b}$ equals the suffix of $(1-\alpha)n$ symbols of $\mathbf{s}$. We represent the codestrings in the codebook using a bipartite graph $G = (V_P, V_S, E)$ with

$$V_P = \left\{\mathbf{a} \in \{0,1\}^{\alpha n} : \exists \mathbf{s} \in C_2^{(p)}(n) \text{ s.t. } \boldsymbol{s} = \mathbf{ab}\right\}, \tag{7}$$

$$V_S = \left\{\mathbf{b} \in \{0,1\}^{(1-\alpha)n} : \exists \mathbf{s} \in C_2^{(p)}(n) \text{ s.t. } \boldsymbol{s} = \mathbf{ab}\right\}. \tag{8}$$

An edge $(v_1, v_2) \in E$, with $v_1 \in V_P$ and $v_2 \in V_S$, connects an admissible prefix (vertex in V_P) to an admissible suffix (vertex in V_S). Hence, an edge corresponds to a codestring in $C_2^{(p)}$ and vice versa. Furthermore, let $w \in \{0,1,\ldots,\alpha n\} = [[\alpha n+1]]$.

We also find it useful to work with another bipartite graph $G^{(w)} = (V_P^{(w)}, V_S^{(w)}, E^{(w)})$ whose edges are a subset of the edges in E. The partition of the vertices $V^{(w)} = (V_P^{(w)}, V_S^{(w)})$ is such that $v_1 \in V_P^{(w)}$ if and only if the prefix $\boldsymbol{a} \in \{0,1\}^{\alpha n}$ represented by the vertex v_1 in G has weight w, and in addition, $v_2 \in V_S^{(w)}$ if and only if there exists a $v_1 \in V_P^{(w)}$ such that $(v_1, v_2) \in E$. The set $E^{(w)} \subseteq E$ is such that $(v_1, v_2) \in E^{(w)}$ if $v_1 \in V_P^{(w)}$ and $v_2 \in V_S^{(w)}$.

Lemma 16: The graph $G^{(w)}$ cannot contain a cycle of length four.

Proof: Suppose to the contrary that $G^{(w)}$ contains a 4-cycle, say $(\mathbf{a}_1\mathbf{b}_1, \mathbf{a}_2\mathbf{b}_2, \mathbf{a}_1\mathbf{b}_2, \mathbf{a}_2\mathbf{b}_1)$. Then, $\mathcal{M}_p(\mathbf{a}_1\mathbf{b}_1) \cup \mathcal{M}_p(\mathbf{a}_2\mathbf{b}_2) = \mathcal{M}_p(\mathbf{a}_2\mathbf{b}_1) \cup \mathcal{M}_p(\mathbf{a}_1\mathbf{b}_2)$. To verify the above claim, note that all prefixes of length αn have to be the same since $\mathcal{M}_p(\mathbf{a}_1) \cup \mathcal{M}_p(\mathbf{a}_2) = \mathcal{M}_p(\mathbf{a}_2) \cup \mathcal{M}_p(\mathbf{a}_1)$. Furthermore, since $\text{wt}(\mathbf{a}_1) = \text{wt}(\mathbf{a}_2)$ it is straightforward to verify that the compositions of all prefixes of length longer than αn are the same in $\mathcal{M}_p(\mathbf{a}_1\mathbf{b}_1) \cup \mathcal{M}_p(\mathbf{a}_2\mathbf{b}_2)$ and $\mathcal{M}_p(\mathbf{a}_2\mathbf{b}_1) \cup \mathcal{M}_p(\mathbf{a}_1\mathbf{b}_2)$. This contradicts the fact that the prefixes and suffixes involved correspond to a 2-prefix code. ■

We are now ready to prove our upper bound on h-prefix codes for $h=2$.

Theorem 17: For any $\epsilon > 0$, there exists an $n_\epsilon > 0$ such that for all $n \geq n_\epsilon$, one has $\frac{1}{n}\log|C_2^{(p)}(n)| \leq \frac{2}{3} + \epsilon$.

Proof: In order to bound the number of codestrings in $C_2^{(p)}(n)$, we will upper bound the number of edges in the graph $G = (V_P, V_S, E)$. To this end, we consider the maximum number of edges in the graph $G^{(w)} = (V_P^{(w)}, V_S^{(w)}, E^{(w)})$. It follows from the pigeonhole principle that there exists a $w^* \in [[\alpha n+1]]$ such that $\left|E^{(w^{(*)})}\right| \geq \frac{|E|}{\alpha n+1}$. Thus, $\frac{1}{n}\log\left|E^{(w^{(*)})}\right|$ can be approximated by $\frac{1}{n}\log\left|C_2^{(p)}(n)\right|$ for n sufficiently large.

According to Lemma 16, $G^{(w^{(*)})}$ cannot contain a 4-cycle. It is known [44] that the number of edges in an $m_1 \times m_2$

bipartite graph without cycles of length 4 is at most

$$m_1 m_2^{\frac{1}{2}} + m_1 + m_2. \tag{9}$$

Letting $\alpha n = \frac{n}{3}$ in (9) so that $m_1 = 2^{n/3}$ and $m_2 = 2^{2n/3}$ gives

$$\frac{1}{n}\log\left|E^{(w^{(*)})}\right| \le \frac{2}{3} + \mathcal{O}\left(\frac{1}{n}\right).$$

■

The next corollary follows from Theorem 17 and Lemma 15.

Corollary 18: A 2-prefix code must have a rate bounded as $R_2^{(MC)} \le \frac{2}{3}$.

Next, we consider the extension to the case where $h > 2$ based on the same approach. Let $C_h^{(p)}(n)$ denote an h-prefix code of length n. As before, we represent our codestrings using a graph $G^{(h)} = (V_P^{(h)}, V_S^{(h)}, E^{(h)})$ as defined in (7) and (8), except that $(\mathbf{a}, \mathbf{b}) \in E^{(h)}$ if and only if $(\mathbf{a}, \mathbf{b}) \in C_h^{(p)}(n)$. As before, we will also work with the bipartite graph $G^{(w,h)} = (V_P^{(w,h)}, V_S^{(w,h)}, E^{(w,h)}) \subseteq G^{(h)}$, which is restricted to only use prefixes of weight w. Lemma 19 is a natural generalization of Lemma 16.

Lemma 19: The graph $G^{(w,h)}$ cannot contain a $2h$-cycle.

Proof: Suppose to the contrary that the statement in the lemma does not hold and that $(\mathbf{a}_1, \mathbf{b}_1)$, $(\mathbf{b}_1, \mathbf{a}_2)$, $(\mathbf{a}_2, \mathbf{b}_2)$, $\ldots$, $(\mathbf{a}_h, \mathbf{b}_h)$, $(\mathbf{b}_h, \mathbf{a}_1)$ forms a 2h-cycle. Then, we have $\mathbf{a}_1\mathbf{b}_1$, $\mathbf{a}_2\mathbf{b}_2$, $\mathbf{a}_3\mathbf{b}_3$, $\ldots$, $\mathbf{a}_h\mathbf{b}_h \in C_h^{(p)}(n)$, as well as $\mathbf{b}_1\mathbf{a}_2, \mathbf{b}_2\mathbf{a}_3, \mathbf{b}_4\mathbf{a}_5, \ldots, \mathbf{b}_h\mathbf{a}_1 \in C_h^{(p)}(n)$. Since all the prefixes in $G^{(w,h)}$ have weight w, the claim follows. ■

Theorem 20: For odd h, $R_h^{(MC)} \le \frac{h+1}{2h}$. For even h, $R_h^{(MC)} \le 1 - \frac{1}{2}\left(\frac{1}{1+\frac{1}{h}}\right)$.

Proof: The result follows using the same arguments as those described in Theorem 17 and Corollary 18 by noting that the maximum number of edges in a $m_1 \times m_2$ bipartite graph that does not contain a cycle of length $2h$ is at most $(m_1 m_2)^{h+1} h + m_1 + m_2$ when h is odd [44]. For the case when h is even, the maximum number of edges equals $m_1^{\frac{k+2}{2k}} m_2^{\frac{1}{2}} + m_1 + m_2$ [44].[2] ■

As a final note, we observe that the work in [27] and [28] also considered the case of nonbinary B_h codes for $h = 2$. The main result is that for a large enough alphabet, the maximum asymptotic rate of nonbinary B_2 codes is at most $\frac{1}{2}$. Furthermore, graph-theoretic methods have used in establishing related bounds for separable codes, reported in [32].

V. A Brief Description of Error Models and Error-Correction

The MS/MS readout technique is error-prone, and not all masses of prefixes and suffixes are measured or reported. Furthermore, polymer fragmentation causes the loss of some atoms and creates errors in the actual mass values. Another type of error occurs when fragmentation fails, in which case both a prefix and suffix of complementary length are missing. A useful assumption for error-correction that we follow is that one can actually determine the length of the prefixes/suffixes based on their masses. This is possible if the masses of 0s and 1s differ significantly (for example, if the masses of the 1 or 0 molecules differ by at least n) or if other design criteria are met.

If at most t_p prefix compositions are erased (missing), and at most t_s suffix compositions are erased (missing), then one needs to correct not more than 2 min $\{t_p, t_s\}$ erasures in the prefix (suffix) string, each occurring in a contiguous burst of length at least two. We show next that one can employ simple one-step or two-step error-control coding approaches to handle missing prefixes/suffixes or instead resort to the use of integrals of strings [11].

We start with a scheme that can correct up to t missing prefix-suffix composition errors. Recall that the B_h codebook $\mathcal{S}_h(n)$, described in Sections II and III, can be constructed using the columns of a parity-check matrix of a code with minimum Hamming distance $d \ge 2h+1$. The idea behind our error-correction technique is to ensure that the real-valued sum of every h-string subset of the code is an error-tolerant codestring. A solution to this problem was proposed in [41] for the purpose of designing signature codes for a noisy MAC (i.e., codes capable of correcting errors in the syndrome of a received word). It consists of encoding the columns of a parity-check matrix $\tilde{H}^{k\times n}$, capable of correcting h substitution errors, using a linear binary code that can correct $\lfloor \frac{t}{2} \rfloor$ substitution errors. Note that the parameter t can be chosen independently from the parameter h as long as $\lfloor \frac{t}{2} \rfloor \le k \le n$. For encoding purposes, the authors suggest using two binary BCH codes, so that $\tilde{H}$ is the parity check matrix of a BCH code of designed distance $\ge 2h+1$, while the parity-check matrix used to introduce error-control redundancy to the columns of $\tilde{H}$ is also chosen according to a BCH code with dimension k, length n and redundancy not exceeding $\lfloor \frac{t}{2} \rfloor \log(n+1)$, capable of correcting at least $\lfloor \frac{t}{2} \rfloor$ substitution errors. Clearly, the only difference is that in our setting, we encounter erasures in the coded strings (the augmented columns), and wish to handle erasures. Note also that this construction, as pointed out by the authors, does not fully exploit the fact that addition is performed over the reals and not over the field $\mathbb{F}_2$.

In [41], one starts with finding the smallest prime $p > h$, and using a linear code over $\mathbb{F}_p$ (e.g., a Reed-Solomon code of length $p-1$) for the syndrome error-control redundancy. The dimension of the latter code equals n, and it is required that the code be able to correct t substitution errors over the field $\mathbb{F}_p$. Since the redundancy is nonbinary, each symbol of the parity-check string is converted into a string of length $\log(p+1)$, representing the binary expansion of the symbol over $\mathbb{F}_p$. The binary expansions are stacked on top of each other according to the given parity-check string. The interesting observation is that, from the sum of the binary strings over the reals, one can clearly obtain the binary expansion of the symbols in the sum, and then generate the residues modulo p of the elements of the string to obtain the redundancy information needed for decoding. The obtained code is linear.

Henceforth, we use the value N to denote the length of the uniquely reconstructable strings ***h*-*MC*** with added error-control redundancy. It is not to be confused with the

[2]Note that the result in our preprint [45] contains an error on page 4. The assertion that the lower and upper bound are asymptotically equal is incorrect.

parameter N from Section III as this particular notation is reused to avoid clutter. Also, as before, we let $\mathcal{S}_h(n)$ be a binary B_h code constructed using the parity-check matrix of a binary code with minimum Hamming distance $\geq 2h+1$; to add the syndrome redundancy, we use a BCH code with appropriate parameters. The main observation is that due to our encoding method, which uses the complementation/bit flipping procedure, we require an unequal error-protection scheme. Recall that the substring $\mathbf{r}$ used in the construction described in Section III is the indicator vector for substring (of length $\sqrt{n}$ bits) flips. Errors in the $\mathbf{r}$ substring may clearly cause a burst of "complementation errors" due to the fact that $\mathbf{r}$ indicates if a string or its complement should be used. There are two approaches one can follow by either encoding the string to handle a larger number of erasures independent on their location (the One-Step procedure) or by adding specialized redundancy to the $\mathbf{r}$ string (the Two-Step procedure).

The One-Step encoding method proceeds as follows:

- Each string $\mathbf{s} \in \mathcal{S}_h(n)$ is encoded using a BCH code into an intermediary string $\mathbf{s}'$ of length m, capable of correcting $t(\sqrt{m}+1)$ erasures. The redundancy required is at most $\lceil \frac{t}{2} \rceil (\sqrt{m}+1)\log(m+1)$.
- The intermediary string $\mathbf{s}'$ of length m is subsequently encoded via the balancing procedure described in Section III. The encoded balanced string has length N and belongs to a h-***MC*** code capable of correcting up to t composition erasures; here, $N = m + \frac{17}{2}\sqrt{m}$, which is at most $n + \lceil \frac{t}{2} \rceil (\sqrt{m}+1)\log(m+1) + \frac{17}{2}\sqrt{m}$; N can be further upper-bounded by

$$n + \lceil \frac{t}{2} \rceil (\sqrt{n}+1)\log n + \frac{17}{2}\sqrt{n} + \epsilon_n \sqrt{n} \times \left(\lceil \frac{t}{2} \rceil \log n + \frac{17}{2}\right) + \lceil \frac{t}{2} \rceil \, \delta_n,$$

where

$$\epsilon_n = \frac{\lceil \frac{t}{2} \rceil (\sqrt{m}+1)\log(m+1)}{2n}$$

and

$$\delta_n = \frac{\lceil \frac{t}{2} \rceil (\sqrt{m}+1)\log(m+1)}{n}.$$

As either the partial prefix-sum or the partial suffix-sum string has $\leq t(\sqrt{m}+1)$ erasures, the binary sum of the input strings can be recovered correctly. The decoding procedure for strings involved in the sum is identical to the one as described in Section III.

We observed in the context of the One-Step scheme that errors in the substring $\mathbf{r}$, encoding information about which blocks are complemented, cause blocks of errors in the global string. Each erasure in $\mathbf{r}$ results in $\sqrt{m}$ additional erasures, where m is the length of the (approximately) balanced substrings. In order to overcome this issue, one can use unequal error-correction schemes that ensure that the binary sum of the $\mathbf{r}$ substring components across the input strings can be recovered independently from the rest of the string. The correctly reconstructed binary $\mathbf{r}$ sum can then be used for subsequent decoding of the complete collection of input strings.

As before, let $\mathcal{S}_h(n)$ be a binary B_h code constructed using the parity-check matrix of a code with minimum Hamming distance $\geq 2h+1$ (say, a BCH code). Furthermore, let N denote the overall length of the h-***MC*** codestrings with added redundancy for mass error-correction. Encoding is performed as follows:

- Each string $\mathbf{s} \in \mathcal{S}_h(n)$ is encoded into an intermediary strings $\mathbf{s}'$ of length m_1 capable of correcting t erasures, using a BCH code. The redundancy is at most $t\log(m_1+1)$, and
$$m_1 \leq n + \lceil \frac{t}{2} \rceil \log(m_1+1).$$
- Each intermediary string $\mathbf{s}'$ of length m_1 is encoded into a Dyck string using the procedure described in Section III, to arrive at a second intermediary string $\mathbf{s}''$ of length m_2, where $m_2 = m_1 + \frac{17}{2}\sqrt{m_1}$.
- The substring $\mathbf{r}$ of the intermediary string $\mathbf{s}''$ is encoded into a codestring $\mathbf{r}\mathbf{r}'$ of total length m_3, capable of correcting t erasures. Let $m_4 = m_3 - \sqrt{m_1}$ denote the length of $\mathbf{r}'$. It is easy to see that $m_4 \leq \lceil \frac{t}{2} \rceil \log(m_3+1)$.
- Since the string has to be balanced, $\mathbf{r}' = r'_1 r'_2 \ldots r'_{m_4}$ is converted into $\mathbf{z} = r'_1 \bar{r'}_1 r'_2 \bar{r'}_2 \ldots r'_{m_4-1} \bar{r'}_{m_4}$, where $\bar{r'}_i = 1 - r'_i$.
- The balanced redundancy $\mathbf{z}$ is appended to the $\mathbf{r}$ substring of the intermediary string $\mathbf{s}''$. Also, a bit 1 is added to the prefix of **1**s and a bit 0 is appended to the suffix of **0**s to preserve the Dyck property of the string.

The length of the coded string equals $N = m_1 + \frac{17}{2}\sqrt{m_1} + 2(m_3 - \sqrt{m_1}) + 2$, and upper-bounded in terms of the length n as

$$n + \lceil \frac{t}{2} \rceil (\log n + \mu_n) + \frac{17}{2}\sqrt{n}(1+\nu_n) + \lceil \frac{t}{2} \rceil (\log n + \mu_n) + 2\theta_n + 2,$$

where

$$\mu_n = \frac{\lceil \frac{t}{2} \rceil \log(m_1+1) + 1}{n},$$
$$\nu_n = \frac{\lceil \frac{t}{2} \rceil \log(m_1+1)}{2n},$$

and

$$\theta_n = \frac{t\log(m_3+1)+1}{\sqrt{n + \lceil \frac{t}{2} \rceil \log(m_1+1)}}.$$

Erasures/errors caused in one mass may result in multiple errors, thereby leading to errors in the reconstructed real-valued sum of the strings. One simple means to mitigate this problem is to use integrals (i.e., running sums) of bits, in which case the errors cancel. Without loss of generality, suppose that $t_p < t_s$. In this case, it is always possible for the errors in the suffix string to be such that we receive no additional information by considering both the prefix and suffix string, and so the problem at hand becomes to recover $\mathbf{s}$ from a set of at most $n - t_p$ prefix compositions.

Claim 21: Suppose that $\mathcal{C}(n,d) \subseteq \mathbb{F}_2^n$ is a code with minimum Hamming distance $d = \min\{t_p, t_s\} + 1$. Let $\mathbf{s} \in \mathbb{F}_2^n$ and fix $wt(\mathbf{s}) = w_0$. Let $\tilde{\mathcal{M}}_p(\mathbf{s})$ be the result of removing t_p compositions from $\mathcal{M}_p(\mathbf{s})$, and t_s compositions from $\mathcal{M}_s(\mathbf{s})$. Then, we can recover $\mathbf{s} = s_1 s_2 \ldots s_n \in \{0,1\}^n$ from $\tilde{\mathcal{M}}_p(\mathbf{s})$

provided that

$$(s_1)\ (s_1+s_2 \mod 2)\ (s_1+s_2+s_3 \mod 2)\ \dots\ (s_1+s_2+\dots+s_n \mod 2) \in \mathcal{C}(n,d).$$

Proof: Without loss of generality, assume that $t_p = \min\{t_p, t_s\}$. The result follows since for $i \in [n]$ the value of the i-th component in the string $(s_1)\ (s_1+s_2 \mod 2)\ (s_1+s_2+s_3 \mod 2)\ \dots\ (s_1+s_2+\dots+s_n \mod 2) \in \mathcal{C}(n,d)$ can be recovered by summing up the number of 1s (modulo 2) in the i-th prefix composition. The claim then follows since we know the lengths of the compositions that are missing from the set $\tilde{\mathcal{M}}_p(\mathbf{s})$ and can hence recover the string $(s_1)\ (s_1+s_2 \mod 2)\ (s_1+s_2+s_3 \mod 2)\ \dots\ (s_1+s_2+\dots+s_n \mod 2)$, where $s \in \mathbb{F}_2^n$, from which $\mathbf{s}$ can be then determined uniquely. Note that for the case that $t_s = \min\{t_p, t_s\}$, since the weight of $\mathbf{s}$ is known, a missing composition of a prefix of length i can be recovered from the known composition of a suffix of length $n-i$. Thus, $t_p + t_s$ missing compositions in $\tilde{\mathcal{M}}_p(\mathbf{s})$ can be recovered from $\tilde{\mathcal{M}}_s(\mathbf{s})$ and w_0. This concludes the proof. ■

Using the result of Claim 21, we can encode our mixtures using the following approach:

- Given a string $\mathbf{s} \in \{0,1\}^n$, construct $I(\mathbf{s}) = (s_1)\ (s_1+s_2 \mod 2)\ (s_1+s_2+s_3 \mod 2)\ \dots\ (s_1+s_2+\dots+s_n \mod 2) = I(\mathbf{s})_1 I(\mathbf{s})_2 \dots I(\mathbf{s})_n$.
- Encode $I(\mathbf{s})$ using a BCH code such that the resulting string $I(\mathbf{s})R'(\mathbf{s})$, where $R'(\mathbf{s})$ is the string of redundancy bits, has length m and can correct $\lfloor \frac{t}{2} \rfloor$ erasures. Observe that $I(\mathbf{s})R'(\mathbf{s})$ does not satisfy the condition in Claim 21.
- Given $R'(\mathbf{s})$, construct $I(\mathbf{s})R(\mathbf{s})$ as follows. First, set $R(\mathbf{s})_1 = R'(\mathbf{s})_1 + I(\mathbf{s})_n$ (modulo 2 addition). Balance the string by setting $R(\mathbf{s})_2 = 1 - R(\mathbf{s})_1$. Next, set $s_1+s_2+\dots+s_n+R(\mathbf{s})_1+R(\mathbf{s})_2+R(\mathbf{s})_3 = R'(\mathbf{s})_2$, which results in $R(\mathbf{s})_3 = I(\mathbf{s})_n + 1 + R'(\mathbf{s})_2$. Similarly, set $R(\mathbf{s})_{2i} = 1 - R(\mathbf{s})_{2i-1}$, and $R(\mathbf{s})_{2i+1} = R'(\mathbf{s})_i + i + I(\mathbf{s})_n$ for all $i \in [m-n-1]$, where m, n are as described in the encoding scheme of Section III.
- Encode $\mathbf{s}$ as $\mathbf{s}R(\mathbf{s})$.

To apply the above procedure, we need to be able to partition the prefix and suffix compositions of the $\mathbf{s}R(\mathbf{s})$. This is easily achieved when $\mathbf{s}R(\mathbf{s})$ is a substring of a Dyck string such that the composition of the prefix preceding the $\mathbf{s}R(\mathbf{s})$-substring in the Dyck string is known. In particular, since the substring $\mathbf{s}R(\mathbf{s})$ occurs after the runlength of 1s in the construction of Section III, the prefix compositions of the constructed string $\mathbf{s}R(\mathbf{s})$ can be recovered by subtracting the weight of the leading runlength of 1s from the corresponding compositions. By construction, $\mathbf{s}R(\mathbf{s})$ satisfies the conditions of Claim 21. Since each code was constructed using a BCH code, the binary sum of multiple strings constructed using this technique also satisfies the conditions in Claim 21.

VI. Open Problems

Many combinatorial and coding-theoretic problems related to string reconstruction from prefix-suffix compositions remain open. A sampling is listed below.

- Our techniques for converting a binary string of length n into strings that are both Dyck and belong to a B_h codebook have suboptimal redundancy. We seek methods that can reduce our overhead and at the same time, offer low encoding and decoding complexity.
- In practice, one often encounters nonbinary alphabets, as polymers can be synthesized to have highly different masses and chemical properties. The question remains to generalize our approach for nonbinary alphabets. Furthermore, it is of interest to investigate such coding techniques for strings that have some form of balanced symbol contents or masses confined to a certain interval.
- It remains an open question to characterize all the missing mass errors that can be corrected by simply utilizing the Dyck, B_h properties of strings and the presence of both prefix and suffix masses.
- At this point, we have no efficient means for correcting mass reducing (or, mass increasing) substitution errors in our mixtures. A solution to this problem can have interesting and important implications in the field of polymer-based data storage.

References

[1] A. Al Ouahabi, J.-A. Amalian, L. Charles, and J.-F. Lutz, "Mass spectrometry sequencing of long digital polymers facilitated by programmed inter-byte fragmentation," *Nature Commun.*, vol. 8, no. 1, pp. 1–8, Dec. 2017.

[2] N. Goldman et al., "Towards practical, high-capacity, low-maintenance information storage in synthesized DNA," *Nature*, vol. 494, no. 7435, p. 77, 2013.

[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," *Angew. Chem. Int. Ed.*, vol. 54, no. 8, pp. 2552–2555, 2015.

[4] S. M. H. T. Yazdi, R. Gabrys, and O. Milenkovic, "Portable and error-free DNA-based data storage," *Sci. Rep.*, vol. 7, no. 1, pp. 1–6, Dec. 2017.

[5] H. T. Yazdi, Y. Yuan, J. Ma, H. Zhao, and O. Milenkovic, "A rewritable, random-access DNA-based storage system," *Sci. Rep.*, vol. 5, pp. 1–10, Sep. 2015.

[6] K. Launay et al., "Precise alkoxyamine design to enable automated tandem mass spectrometry sequencing of digital poly(phosphodiester)s," *Angew. Chem. Int. Ed.*, vol. 60, no. 2, pp. 917–926, Jan. 2021.

[7] S. K. Tabatabaei et al., "DNA punch cards for storing data on native DNA sequences via enzymatic nicking," *Nature Commun.*, vol. 11, no. 1, pp. 1–10, Dec. 2020.

[8] C. Pan, S. K. Tabatabaei, S. M. H. Tabatabaei Yazdi, A. G. Hernandez, C. M. Schroeder, and O. Milenkovic, "Rewritable two-dimensional DNA-based data storage with machine learning reconstruction," *Nature Commun.*, vol. 13, no. 1, Dec. 2022.

[9] H. M. Kiah, G. J. Puleo, and O. Milenkovic, "Codes for DNA sequence profiles," *IEEE Trans. Inf. Theory*, vol. 62, no. 6, pp. 3125–3146, Jun. 2016.

[10] R. Gabrys, H. M. Kiah, and O. Milenkovic, "Asymmetric Lee distance codes for DNA-based storage," *IEEE Trans. Inf. Theory*, vol. 63, no. 8, pp. 4982–4995, Aug. 2017.

[11] R. Gabrys, E. Yaakobi, and O. Milenkovic, "Codes in the Damerau distance for deletion and adjacent transposition correction," *IEEE Trans. Inf. Theory*, vol. 64, no. 4, pp. 2550–2570, Apr. 2018.

[12] R. Gabrys and O. Milenkovic, "Unique reconstruction of coded strings from multiset substring spectra," *IEEE Trans. Inf. Theory*, vol. 65, no. 12, pp. 7682–7696, Dec. 2019.

[13] R. Gabrys, H. S. Dau, C. J. Colbourn, and O. Milenkovic, "Set-codes with small intersections and small discrepancies," *SIAM J. Discrete Math.*, vol. 34, no. 2, pp. 1148–1171, Jan. 2020.

[14] A. Agarwal, O. Milenkovic, S. Pattabiraman, and J. Ribeiro, "Group testing with runlength constraints for topological molecular storage," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2020, pp. 132–137.

[15] M. Cheraghchi, R. Gabrys, O. Milenkovic, and J. Ribeiro, "Coded trace reconstruction," *IEEE Trans. Inf. Theory*, vol. 66, no. 10, pp. 6084–6103, May 2020.

[16] S. Jain, F. Farnoud, M. Schwartz, and J. Bruck, "Duplication-correcting codes for data storage in the DNA of living organisms," *IEEE Trans. Inf. Theory*, vol. 63, no. 8, pp. 4996–5010, Aug. 2017.

[17] N. Raviv, M. Schwartz, and E. Yaakobi, "Rank-modulation codes for DNA storage with shotgun sequencing," *IEEE Trans. Inf. Theory*, vol. 65, no. 1, pp. 50–64, Jan. 2019.

[18] M. Abroshan, R. Venkataramanan, L. Dolecek, and A. G. I. Fabregas, "Coding for deletion channels with multiple traces," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2019, pp. 1372–1376.

[19] A. Lenz, P. H. Siegel, A. Wachter-Zeh, and E. Yaakobi, "Anchor-based correction of substitutions in indexed sets," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2019, pp. 757–761.

[20] Z. Chang, J. Chrisnata, M. F. Ezerman, and H. M. Kiah, "Rates of DNA sequence profiles for practical values of read lengths," *IEEE Trans. Inf. Theory*, vol. 63, no. 11, pp. 7166–7177, Nov. 2017.

[21] I. Shomorony and R. Heckel, "DNA-based storage: Models and fundamental limits," *IEEE Trans. Inf. Theory*, vol. 67, no. 6, pp. 3675–3689, Jun. 2021.

[22] J. Acharya, H. Das, O. Milenkovic, A. Orlitsky, and S. Pan, "String reconstruction from substring compositions," 2014, *arXiv:1403.2439*.

[23] S. Pattabiraman, R. Gabrys, and O. Milenkovic, "Reconstruction and error-correction codes for polymer-based data storage," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Visby, Sweden, Aug. 2019, pp. 1–5.

[24] R. Gabrys, S. Pattabiraman, and O. Milenkovic, "Mass error-correction codes for polymer-based data storage," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Los Angeles, CA, USA, Jun. 2020, pp. 25–30.

[25] S. Pattabiraman, R. Gabrys, and O. Milenkovic, "Coding for polymer-based data storage," 2020, *arXiv:2003.02121*.

[26] B. Lindstrom, "Determining subsets by unramified experiments," in *A Survey of Statistical Design and Linear Models*. Amsterdam, The Netherlands: North Holland, 1975.

[27] B. Lindström, "Determination of two vectors from the sum," *J. Combinatory Theory*, vol. 6, no. 4, pp. 402–407, 1969.

[28] B. Lindström, "On B2-sequences of vectors," *J. Number Theory*, vol. 4, no. 3, pp. 261–265, Jun. 1972.

[29] H. V. Tilborg, "An upper bound for codes in a two-access binary erasure channel (corresp.)," *IEEE Trans. Inf. Theory*, vol. IT-24, no. 1, pp. 112–116, Jan. 1978.

[30] A. G. D'yachkov and V. V. Rykov, "Bounds on the length of disjunctive codes," *Problemy Peredachi Inf.*, vol. 18, no. 3, pp. 7–13, 1982.

[31] A. G. D'yachkov, "Lectures on designing screening experiments," 2014, *arXiv:1401.7505*.

[32] A. D'yachkov, N. Polyanskii, V. Shchukin, and I. Vorobyev, "Separable codes for the symmetric multiple-access channel," *IEEE Trans. Inf. Theory*, vol. 65, no. 6, pp. 3738–3750, Jun. 2019.

[33] A. D'yachkov and V. Rykov, "On a lower bound to the length of B4-codes," *Problems Control Inf. Theory*, vol. 10, no. 5, pp. 301–307, 1981.

[34] S. I. Bross and I. F. Blake, "Upper bound for uniquely decodable codes in a binary input N-user adder channel," *IEEE Trans. Inf. Theory*, vol. 44, no. 1, pp. 334–340, Jan. 1998.

[35] Y. Gu, "Zero-error communication over adder MAC," 2018, *arXiv:1809.07364*.

[36] H. Halberstam and K. F. Roth, *Sequences*. Berlin, Germany: Springer, 2012.

[37] K. O'Bryant, "A complete annotated bibliography of work related to Sidon sequences," 2004, *arXiv:math/0407117*.

[38] J. Singer, "A theorem in finite projective geometry and some applications to number theory," *Trans. Amer. Math. Soc.*, vol. 43, no. 3, pp. 377–385, May 1938.

[39] G. Cohen, S. Litsyn, and G. Zémor, "Binary B2-sequences : A new upper bound," *J. Combinat. Theory A*, vol. 94, no. 1, pp. 152–155, Apr. 2001.

[40] V. Grebinski and G. Kucherov, "Optimal reconstruction of graphs under the additive model," *Algorithmica*, vol. 28, no. 1, pp. 104–124, Sep. 2000.

[41] V. Gritsenko, G. Kabatiansky, V. Lebedev, and A. Maevskiy, "On codes for multiple access adder channel with noise and feedback," in *Proc. 9th Int. Workshop Coding Cryptogr.*, 2015, pp. 1–8.

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

[43] B. Lindström, "On a combinatory detection problem I," *I. Magyar Tud. Akad. Mat. Kutató Int. Közl*, vol. 9, pp. 195–207, Sep. 1964.

[44] A. Naor and J. Verstraëte, "A note on bipartite graphs without 2K-cycles," *Combinatorics Probab. Comput.*, vol. 14, no. 5, pp. 845–849, 2005.

[45] R. Gabrys, S. Pattabiraman, and O. Milenkovic, "Reconstructing mixtures of coded strings from prefix and suffix compositions," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Apr. 2021, pp. 1–5.

Ryan Gabrys (Member, IEEE) received the B.S. degree in mathematics and computer science from the University of Illinois at Urbana–Champaing in 2005, and the Ph.D. degree in electrical engineering from the University of California, Los Angeles, in 2014. He is currently a Scientist jointly affiliated with the Naval Information Warfare Center and the California Institute for Telecommunications and Information Technology (Calit2) at the University of California, San Diego. His research interests include theoretical computer science and electrical engineering, including coding theory, combinatorics, and communication theory.

Srilakshmi Pattabiraman (Student Member, IEEE) received the B.Tech. degree in instrumentation and control engineering from the National Institute of Technology, Trichy, India, in 2015, the master's degree in electrical and computer engineering from The University of Texas at Austin in December 2017, and the Ph.D. degree in electrical and computer engineering from the University of Illinois Urbana–Champaign, in December 2021. She is drawn toward mathematical challenges in combinatorics and coding theory, information theory, detection and estimation theory, and statistical learning theory.

Olgica Milenkovic (Fellow, IEEE) received the M.Sc. degree in mathematics and the Ph.D. degree in electrical engineering from the University of Michigan, Ann Arbor, in 2001 and 2002, respectively. She is the Franklin W. Woeltge Professor of electrical and computer engineering with the University of Illinois Urbana–Champaign (UIUC). Her scholarly contributions have been recognized by multiple awards, including the NSF Faculty Early Career Development (CAREER) Award, the DARPA Young Faculty Award, the Deans Excellence in Research Award, and several Best Paper Awards. In 2013, she was elected a UIUC Center for Advanced Study Associate and a Willett Scholar, while in 2015, she was a elected Distinguished Lecturer of the Information Theory Society. She was an Associate Editor of the IEEE TRANSACTIONS OF COMMUNICATIONS, the IEEE TRANSACTIONS ON SIGNAL PROCESSING, the IEEE TRANSACTIONS ON INFORMATION THEORY, and the IEEE TRANSACTIONS ON MOLECULAR, BIOLOGICAL AND MULTI-SCALE COMMUNICATIONS. In 2009 and 2020, she was the Guest Editor in Chief of special issues of the IEEE TRANSACTIONS ON INFORMATION THEORY.