Learning-Augmented Streaming Codes are Approximately Optimal for Variable-Size Messages

Michael Rudow and K.V. Rashmi

2022 IEEE International Symposium on Information Theory (ISIT) | 978-1-6654-2159-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/ISIT50566.2022.9834539

***Abstract*—Real-time streaming communication requires a high quality-of-service despite contending with packet loss. Streaming codes are a class of codes best suited for this setting. A key challenge for streaming codes is that they operate in an "online" setting in which the amount of data to be transmitted varies over time and is not known in advance. Mitigating the adverse effects of variability requires spreading the data that arrives at a time slot over multiple future packets, and the optimal strategy for spreading depends on the arrival pattern. Algebraic coding techniques alone are therefore insufficient for designing rate-optimal codes. We combine algebraic coding techniques with a learning-augmented algorithm for spreading to design the first approximately rate-optimal streaming codes for a range of parameter regimes that are important for practical applications.**

An extended version of this paper is available at: [1].

I. Introduction

Real-time communication arises in many popular applications, including VoIP, online gaming, and videoconferencing. These applications involve a sender transmitting packets of information to a receiver over a lossy channel. The receiver must decode the data within a strict playback deadline. In many scenarios, one cannot retransmit the lost packets because doing so requires an extra round trip time and can thus exceed the maximum tolerable latency [2]. Instead, one can use erasure coding to recover lost packets.

While erasure coding has been well studied, real-time communication has several unique aspects that require a new "streaming model," as was introduced by Martinian and Sundberg [3]. During each time slot, i, a "message packet," denoted as $S[i]$, of size k arrives at a sender. The sender then transmits a "channel packet," denoted as $X[i]$, of size n to a receiver over a burst-only loss channel. The sender must recover $S[i]$ by time slot $(i+\tau)$. An overview of the model is presented in Figure 1, with the sender, channel, and receiver appearing in blue. (The component "side information" is introduced later.) Coding schemes that recover lost symbols from each message packet τ time slots later have significantly higher rates than alternatives, such as interleaved maximal distance separable codes, that recover all lost symbols together by τ time slots after the *first* message packet for which the symbols are lost [3].

Numerous works [4]–[18] have employed tools from algebraic coding theory to design optimal streaming codes for various settings where the sizes of message packets and channel packets are fixed in advance. These regimes are suitable for applications that involve sending fixed quantities of data, such as VoIP when audio packets are sent uncompressed.

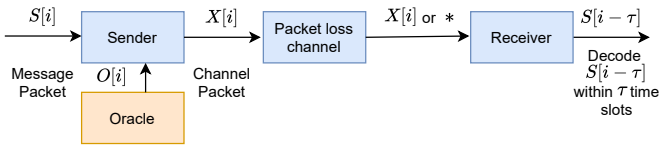

Fig. 1: Overview of the model for streaming codes.

In contrast, many applications, such as videoconferencing, involve transmitting *variable* amounts of data. A new streaming model incorporating message packets of varying sizes was introduced in [19]. Two factors affect the optimal rate in this setting. First is the sequence of the sizes of all message packets of a transmission, called the "message packet size sequence." Second is the maximum number of time slots the receiver can wait to decode each message packet under a lossless transmission, called "lossless delay" and denoted as τ_L.

The optimal rate for "offline" schemes (which know the message packet size sequence) exceeds that of "online" schemes (which cannot access the sizes of message packets for future time slots) in all but two settings [20]. In one setting where τ_L has its maximum possible value, a technique for spreading the symbols of each message packet over several channel packets independently of all other message packets is rate optimal. The other setting, where τ_L has its minimum possible value (i.e., 0), requires sending the symbols of each message packet in the corresponding channel packet. Therefore, information about the sizes of the future message packets does not help. Rate-optimal constructions, or even approximately rate-optimal constructions, are not known even for the offline setting for all remaining parameter regimes.

We consider the setting of $\tau_L = 1$, which is important since it is the smallest value for which the symbols of a message packet can be spread over *multiple* channel packets. Spreading helps to significantly mitigate the adverse effect of variability of the sizes of message packets on the rate [19]. On the other hand, maintaining a small value of τ_L is crucial for latency-sensitive applications, as the delay of τ_L extra time slots may be incurred for decoding *every* message packet.

We first consider the offline setting and decompose the code design into two distinct challenges. First, how can we best spread message symbols over channel packets? Second, how can we send the minimum necessary number of parity symbols to ensure that each message packet is decoded in time, given any choice of how to spread message symbols? We use an integer program offline to determine how to optimally spread message symbols over channel packets. We then introduce a building block for constructing a rate-optimal streaming code

for *any given* choice of how to spread message symbols over channel packets. One final challenge remains: how can we construct a rate-optimal *online* streaming code?

We address the problem by *combining machine learning with tools from algebraic coding theory.* We take a learning-based approach, relying on a technique similar to empirical risk minimization to convert the optimal offline solution into an approximately optimal online one that maximizes the expected rate. Our proposed method determines how to spread symbols online, and then the building block construction is applied.

Our methodology can be viewed as using a "learning-augmented algorithm"—a topic that has recently surged to prominence, tackling problems in other domains, such as caching [21], metric task systems [22], bloom filters [23], learned index structures [24], scheduling [25], etc. [26]–[30]. However, to the best of our knowledge, the powerful paradigm of learning-augmented algorithms has not been applied to design coding schemes until now.

Using machine learning models to perform encoding and/or decoding has received considerable attention in the recent past, including for channel coding [31]–[39], decoding with feedback [40], [41], approximate coded computation [42], [43], MIMO [44], [45], etc. [46]–[49]. Most of these works use neural networks to handle encoding and or decoding in a black-box manner. In contrast, our work applies machine learning only to a small portion of the problem: to determine how to spread message symbols to address the uncertainty in the sizes of the future message packets. We then leverage tools from classical algebraic coding theory to solve the rest of the problem. This makes the learning part lightweight and interpretable and allows for theoretical guarantees on the rate.

II. Model and background

We now present the system model used in this work, which is built on top of the model of streaming codes for variable-size messages [19], [20].

A transmission occurs over $(t+1)$ time slots for a positive integer t. During the ith time slot for $i \in \{0, \ldots, t\}$, the sender obtains a message packet, $S[i] \in \mathbb{F}^{k_i}$, where $\mathbb{F}$ is a finite field, and $k_i \in \{0, \ldots, m\}$ for a positive integer m. The sender also receives "side information," $O[i]$, that captures the differences between the online and offline settings. In the offline setting, which assumes knowledge of the future, the side information is the sizes of the future message packets. In the online setting, the side information is $\mathcal{S}$ independent samples from the distribution of the sizes of future message packets for a positive integer $\mathcal{S}$. Let $D_{k_0,\ldots,k_i}$ be the conditional distribution of $k_{i+1}, \ldots, k_t$ given $k_0, \ldots, k_i$. Then,

$$O[i] = \begin{cases} (k_{i+1}, \ldots, k_t) & \text{if offline} \\ \left\langle \left(k_{i+1}^{(j)}, \ldots, k_t^{(j)}\right) \sim D_{k_0,\ldots,k_i} \right. & \\ \left. \quad \mid j \in \{0, \ldots, \mathcal{S}-1\} \right\rangle & \text{if online.} \end{cases} \tag{1}$$

During the ith time slot, the sender transmits a channel packet, $X[i] \in \mathbb{F}^{n_i}$ to a receiver, where n_i is a non-negative integer. The receiver obtains $Y[i] \in \{X[i], *\}$, which reflects packet reception or packet loss, respectively. When all channel packets are received, $S[i]$ must be decoded by the receiver by time slot $(i + \tau_L)$ for a parameter $\tau_L \geq 0$. This requirement is called the "lossless-delay" constraint and represents the maximum tolerable latency for lossless transmission. Recall from Section I that our work considers $\tau_L = 1$. When losses occur, $S[i]$ must be decoded by time slot $(i+\tau)$ for a parameter $\tau \geq \tau_L$. This reflects the maximum acceptable latency in the worst case and is called the "worst-case-delay" constraint.

Our work uses the packet loss model from previous work [3], [4], [20]. Packets are lost in bursts of up to b consecutive losses followed by at least τ successful receptions. Formally, for any $i \in \{0, \ldots, t-\tau-b+1\}$, if $(Y[i] = *)$ then $\forall j \in \{i+b, \ldots, i+b+\tau-1\}$ $(Y[j] = X[j])$. To ensure that the worst-case-delay constraint is satisfiable, $\tau \geq b$. Furthermore, $b > 0$ because coding is not needed otherwise. Finally, $\tau > b$, since $\tau \geq (\tau_L + b)$ [19] and $\tau_L = 1$ in our work.

When the sizes of message packets and channel packets are fixed, the rate is simply the ratio of their sizes. However, a more nuanced notion of rate is needed due to the variability in the sizes of message packets and channel packets. The rate for a message packet size sequence $k_0, \ldots, k_t$ is defined as

$$R_t = \frac{\sum_{i=0}^{t} k_i}{\sum_{i=0}^{t} n_i}. \tag{2}$$

For any integer i, $\{0, \ldots, i\}$ is denoted by $[i]$. For convenience of notation, t is taken to be at least 4τ, and $k_i = 0$ for $i \in [2\tau - 1] \cup \{t - 2\tau + 1, \ldots, t\}$. This assumption is satisfied by adding zero padding, which does not affect the rate. Encoding and decoding depends on the history of the transmission, which is summarized as follows.

Definition 1 (State): For any t, τ, and $i \in \{2\tau, \ldots, t\}$, the *state* is denoted $\mathcal{X}_i = (k_0, \ldots, k_i, X[0], \ldots, X[i-1])$.

This section considers systematic codes for clarity, but the results also hold for general codes. To meet the lossless-delay constraint, the symbols of $S[i]$ must be sent by time slot $(i + \tau_L)$ (i.e., in $X[i]$ and $X[i+1]$). The "policy" of a construction, as defined below, specifies how to spread the message symbols.

Definition 2 (Policy): The *policy* of a construction for any $i \in [t]$ and state $\mathcal{X}_i$ is the number of symbols of $S[i]$ sent in channel packet $X[i]$. The policy is denoted as $\mathcal{F}_i(\mathcal{X}_i)$ (or f_i for conciseness) and lies in $[k_i]$.

For any $i > 0$, $X[i]$ comprises (a) the first f_i symbols of $S[i]$, (b) the final $(k_{i-1} - f_{i-1})$ symbols of $S[i-1]$, and (c) p_i parity symbols, denoted as $P[i]$. The encoding is given by $X[i] =$

$$Enc(\mathcal{X}_i, S[i-\tau], \ldots, S[i-1], S_0[i], \ldots, S_{f_i-1}[i], O[i]) \tag{3}$$

for $i \geq 2\tau$, and $X[i]$ is empty for $i < 2\tau$. This section assumes that $X[i]$ is independent of the message symbols of $S[i]$ sent in $X[i+1]$ for clarity, although the results hold without this assumption. The receiver obtains $Y[i] \in \{X[i], *\}$ depending on whether channel packet $X[i]$ is received or dropped.[1]

[1]The receiver needs the sizes of the message packets to decode. Thus, a small header with up to $2\sum_{j=i-b}^{i} \log(k_j) \leq 2(b+1)\log(m)$ symbols containing $(k_{i-b}, \ldots, k_i)$ is added to the header of $X[i]$.

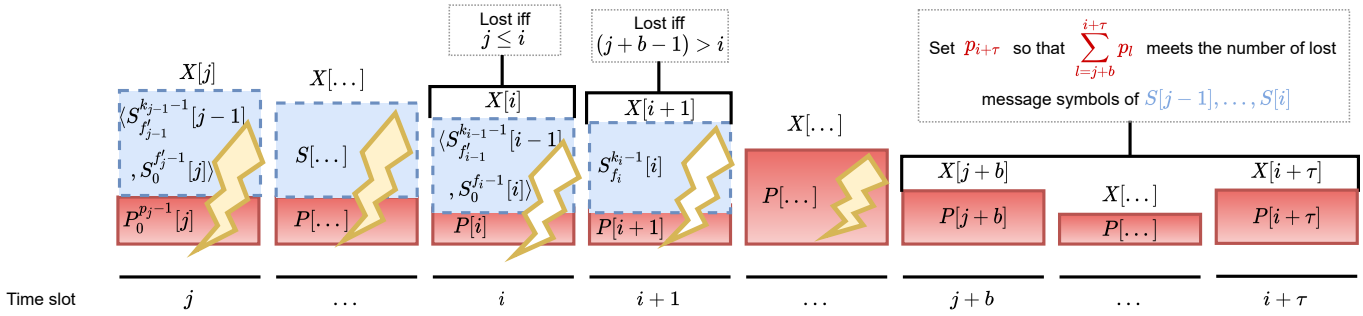

Fig. 2: Selecting $p_{i+\tau}$ by considering each burst starting in time slot $j \in \{i-b+1, \ldots, i+1\}$ (shown with lightning bolts). Under lossless transmission, $S[i]$ is available in uncoded form. Otherwise, $S[i]$ is decoded as

$$Dec\left(\langle Y[j], k_j, f_j \mid j \in [i+\tau]\rangle\right). \tag{4}$$

The following notation is used throughout our work. A vector V has length v, comprises symbols $(V_0, \ldots, V_{v-1})$, and is a row vector. For any $i \le j \in [v-1]$, $V_i^j = (V_i, \ldots, V_j)$.

III. A Building Block Construction

In this section, we present a rate-optimal construction, called the "$(\tau, b, t, \langle f_i \mid i \in [t]\rangle)$ −Spread Code," for any given policies, i.e., choice of how to spread the message symbols over channel packets. Specifically, for any given $\langle f_i \mid i \in [t]\rangle$, at least f_i symbols of $S[i]$ will be sent in $X[i]$ for each $i \in [t]$.

The first 2τ channel packets are empty. For each $i \in [t] \setminus [2\tau-1]$, $X[i]$ comprises (a) the first f'_i symbols of $S[i]$ for some $f'_i \ge f_i$, (b) the final $(k_{i-1} - f'_{i-1})$ symbols of $S[i-1]$, and (c) p_i parity symbols called $P[i]$. Next, we define f'_i, p_i, and $P[i]$ for any message packet size sequence, $k_0, \ldots, k_t$.

Defining each f'_i and p_i. For $i \in [2\tau-1] \cup \{t-2\tau+1, \ldots, t\}$, $f'_i = k_i = 0$. For $i \in [3\tau-1] \cup \{t-\tau+1, \ldots, t\}$, $p_i = 0$. For all $i = 2\tau, \ldots, (t-2\tau)$, we define $p_{i+\tau}$ to be as small as possible while ensuring that $S[i]$ is decoded by time slot $(i+\tau)$ under any lossy transmission. Specifically, $p_{i+\tau} =$

$$\max_{j \in \{i-b+1, \ldots, i+1\}} \Big(0, \mathbb{1}[j+b-1 \ge i+1]\,(k_i - f_i) + \mathbb{1}[j \le i] f_i + \sum_{l=j}^{i} (k_{l-1} - f'_{l-1}) + \sum_{l=j}^{i-1} f'_l - \sum_{l=j+b}^{i+\tau-1} p_l\Big), \tag{5}$$

as is illustrated in Figure 2. We then use $p_{i+\tau}$ to define

$$f'_i = \max\left(f_i, p_{i+\tau}\right). \tag{6}$$

Constructing parity symbols. The parity symbols are defined analogously to those of the construction from [20] (which builds on the construction from [5]). For $i \in \{2\tau, \ldots, t-\tau\}$, the message symbols sent in $X[i]$ are partitioned into (a) symbols of $S[i]$ that are recovered during time slot $(i+\tau)$ under a lossy transmission, and (b) symbols of $S[i-1]$ and $S[i]$ that are recovered by time slot $(i-1+\tau)$ under a lossy transmission. The two components are of sizes u_i and v_i and are denoted as $U[i]$ and $V[i]$, respectively. Thus, $X[i] = (U[i], V[i], P[i])$, where

$$U[i] = S_0^{p_{i+\tau}-1}[i] \tag{7}$$

$$V[i] = \left(S_{p_{i+\tau}}^{k_i - f'_i - 1}[i], S_{f'_{i-1}}^{k_{i-1}-1}[i-1]\right) \tag{8}$$

$$P[i] = U[i-\tau] + P^{(v)}[i]. \tag{9}$$

Each symbol of $P^{(v)}[i]$ is a linear combination of the symbols of $(V[i-\tau], \ldots, V[i-1])$, where the linear equations are chosen using a $(2m\tau) \times (2m\tau)$ Cauchy matrix, A, as follows. Let $W[i]$ be a length $2m\tau$ vector where positions $2m(j \bmod \tau), \ldots, (2m(j \bmod \tau) + 2m - 1)$ comprise $V[j]$ followed by $(2m - v_j)$ 0's for $j \in \{i-\tau, \ldots, i-1\}$, as is illustrated in Figure 3. Finally, we define $P^{(v)}[i] = W[i]A_{(i)}$, where $A_{(i)}$ is A restricted to columns $2m(i \bmod \tau), \ldots, (2m(i \bmod \tau) + p_i - 1)$.

Decoding. For $i \in [t]$, $S[i]$ is decoded (a) from $X[i]$ and $X[i+1]$ if there are no losses, and (b) by solving a system of linear equations corresponding to the symbols of $S[i-\tau], \ldots, S[i-1], Y[i], \ldots, Y[i+\tau]$ after a burst (shown in Figure 4).

Next, we show that the $(\tau, b, t, \langle f_i \mid i \in [t]\rangle)$ −Spread Code meets the lossless-delay and worst-case-delay constraints.

Lemma 1: For any parameters (τ, b) and message packet size sequence $k_0, \ldots, k_t$, and policy f_i for $i \in [t]$, the $(\tau, b, t, \langle f_i \mid i \in [t]\rangle)$ −Spread Code satisfies the lossless-delay constraint and worst-case-delay constraint.

Proof sketch: The lossless-delay constraint is met by sending $S[i]$ over $X[i]$ and $X[i+1]$. For any burst of length b starting in time slot i, $P[i+b], \ldots, P[i+\tau-1]$ are used to recover $V[i], \ldots, V[i+b-1]$. Then $U[j]$ is recovered using $P[j+\tau]$ for $j \in \{i, \ldots, i+b-1\}$. As such, $S[i], \ldots, S[i+b-1]$ are recovered by time slots $(i+\tau), \ldots, (i+b-1+\tau)$ respectively. A complete proof is included in [1]. ∎

Next, we provide a lower bound on the number of parity symbols sent by any streaming code that satisfies the lossless-delay and worst-case-delay constraints.

Lemma 2: Consider any τ, t, b, and any streaming code that satisfies the lossless-delay and worst-case-delay constraints. Suppose for $l \in [t]$, the construction sends $p_l^\dagger$ parity symbols and uses policy f_l. For any $i \in \{3\tau, \ldots, t\}$ and $j \in \{i-\tau-b+1, \ldots, i-\tau+1\}$, the number of parity symbols satisfies

$$-f_{j-1} - \mathbb{1}[j+b-1 = i-\tau]\left(k_{i-\tau} - f_{i-\tau}\right) + \sum_{l=j-1}^{i-\tau} k_l \le \sum_{l=j+b}^{i} p_l^\dagger. \tag{10}$$

Proof sketch: Suppose $X[j], \ldots, X[j+b-1]$ are lost. Due to the worst-case-delay constraint, $S[j-1], \ldots, S[i-\tau]$ must be recovered by time slot i. Thus, $\sum_{l=j-1}^{i-\tau} k_l$ message symbols must be decoded, while f_{j-1} symbols of $S[j-1]$ are received in $X[j-1]$ and, if $X[i-\tau+1]$ is received,

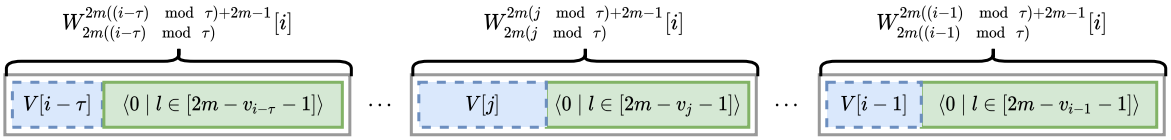

Fig. 3: Defining $W[i]$ by placing the symbols of $V[j]$ in positions $2m(j \bmod \tau), \ldots, (2m(j \bmod \tau) + v_j - 1)$ for $j \in \{i - \tau, \ldots, i-1\}$. The remaining positions are filled with 0's.

$\left(k_{i-\tau} - f_{i-\tau}\right)$ symbols of $S[i-\tau]$ are received. By the independence of message packets, the remaining message symbols received in $X[j+b], \ldots, X[i]$ contain no information about $S[j-1], \ldots, S[i-\tau]$. Enough parity symbols must be received in $X[j+b], \ldots, X[i]$ to recover the lost symbols of $S[j-1], \ldots, S[i-\tau]$. A complete proof is included in [1]. ■

We show the rate of the $(\tau, b, t, \langle f_i \mid i \in [t] \rangle)$ −Spread Code matches that of any streaming code with policy f_i for $i \in [t]$.

Lemma 3: For any τ, t, b, and message packet size sequence $k_0, \ldots, k_t$, the $(\tau, b, t, \langle f_i \mid i \in [t] \rangle)$ −Spread Code matches the rate of any streaming code with policy f_i for $i \in [t]$ that satisfies the lossless-delay and worst-case-delay constraints.

Proof sketch: Under the $(\tau, b, t, \langle f_i \mid i \in [t] \rangle)$ −Spread Code, $\sum_{l=0}^{t}(k_l + p_l)$ symbols are sent. Consider any streaming code that satisfies the lossless-delay and worst-case-delay constraints , and, for each $i \in [t]$, employs policy f_i and sends $p_i^\dagger$ parity symbols. The code sends $\sum_{l=0}^{t}\left(k_l + p_l^\dagger\right)$ symbols.

We show by induction on $i = 0, \ldots, t$ that $\sum_{l=0}^{i} p_l \leq \sum_{l=0}^{i} p_l^\dagger$. The base case holds for $l < 3\tau$ because $p_0 = 0, \ldots, p_{3\tau-1} = 0$. In the inductive hypothesis, for all $j < i$:

$$\sum_{l=0}^{j} p_l \leq \sum_{l=0}^{j} p_l^\dagger. \tag{11}$$

The inductive step for $i \geq 3\tau$ holds when $p_i \leq p_i^\dagger$. Otherwise, there is a burst starting in $j_* \in \{i-\tau-b+1, \ldots, i-\tau+1\}$ so that the number of parity symbols sent over $X[j_* + b], \ldots, X[i]$ (i.e., $\sum_{l=j_*+b}^{i} p_l^\dagger$) is at least $\sum_{l=j_*+b}^{i} p_l$. Combining this with Eq. (11) (i.e., $\sum_{l=0}^{j_*+b-1} p_l \leq \sum_{l=0}^{j_*+b-1} p_l^\dagger$) concludes the proof. A complete proof is included in [1]. ■

IV. Offline-optimal streaming codes

In this section, we design the first rate-optimal offline construction for the setting of $\tau_L = 1$. We build the construction in two steps for an arbitrary message packet size sequence, $k_0, \ldots, k_t$. First, we design an integer program (IP) to use constraints to model satisfying the lossless-delay constraint and Lemma 2. The IP determines an optimal policy for each time slot (i.e., $\langle f_i \mid i \in [t] \rangle$) by setting the objective function to minimize the total number of parity symbols transmitted, which maximizes the rate. Second, we employ the $(\tau, b, t, \langle f_i \mid i \in [t] \rangle)$ −Spread Code given the polices.

Next, we introduce Algorithm 1 to determine an optimal policy, f_i, for each time slot $i \in [t]$ and then verify that the $(\tau, b, t, \langle f_i \mid i \in [t] \rangle)$ −Spread Code is rate optimal.

Theorem 1: For any (τ, b, t) and message packet size sequence $k_0, \ldots, k_t$, suppose Algorithm 1 outputs $\langle f_i \mid i \in [t] \rangle$. Then the $(\tau, b, t, \langle f_i \mid i \in [t] \rangle)$ −Spread Code is rate optimal.

Algorithm 1 Computes $\langle f_i \mid i \in [t] \rangle$ leading to an offline rate-optimal $(\tau, b, t, \langle f_i \mid i \in [t] \rangle)$ −Spread Code.

Input: $(\tau, b, t, k_0, \ldots, k_t)$

Minimize $\sum_{i=0}^{t} p_i^{(IP)}$ subject to

- $\forall i \in [t]$, $f_i^{(IP)} \geq 0, f_i^{(IP)} \leq k_i, p_i^{(IP)} \geq 0$.
- $\forall i \in \{3\tau, \ldots, t-\tau\}, j \in \{i-\tau-b+1, \ldots, i-\tau+1\}$,

$$- f_{j-1}^{(IP)} - \mathbb{1}[j+b-1 = i-\tau]\left(k_{i-\tau} - f_{i-\tau}^{(IP)}\right) + \sum_{l=j-1}^{i-\tau} k_l \leq \sum_{l=j+b}^{i} p_l^{(IP)}.$$

Output: $\langle f_i \mid i \in [t] \rangle$

Proof sketch: At a high level, no scheme can send fewer than $\sum_{i=0}^{t} p_i^{(IP)}$ parity symbols (Lemma 2 and the minimality of the IP), and the $\left(\tau, b, t, \left\langle p_i^{(IP)} \mid i \in [t] \right\rangle\right)$ −Spread Code sends at most $\sum_{i=0}^{t} p_i^{(IP)}$ parity symbols (Lemma 3). A complete proof is included in [1]. ■

Although Algorithm 1 applies to the entire message packet size sequence, it is trivial to modify the algorithm to apply to the remainder of a transmission after channel packets $X[0], \ldots, X[l]$ have been sent for an arbitrary $l \in [t]$. This involves adding constraints for all $j \in [l]$ (a) $f_j^{(IP)} = f_j$ and (b) $p_j^{(IP)} = p_j$. We call the modified algorithm "Algorithm 2."

Corollary 1: For any (τ, b, t), message packet size sequence $k_0, \ldots, k_t$, and $l \in [t]$, suppose that for all $j \in [l]$, policy f_j was used and p_j parity symbols were sent in $X[j]$, and Algorithm 2 outputs $\langle f_i \mid i \in [t] \rangle$. Then the $(\tau, b, t, \langle f_i \mid i \in [t] \rangle)$ −Spread Code attains the best possible rate given the prior transmission of $X[0], \ldots, X[l]$.

V. Learning-based online streaming codes

We now present an online code construction, dubbed the "(τ, b, t)−Spread ML Code," whose expected rate is within ϵ of the online-optimal-rate. The construction uses a learning-based approach to specify the policy for spreading the symbols of $S[i]$, denoted $f_i^{(\epsilon)}$, for each $i \in [t]$, and then applies the $\left(\tau, b, t, \left\langle f_i^{(\epsilon)} \mid i \in [t] \right\rangle\right)$ −Spread Code .

Recall from Eq. (1) that the side information is $\mathcal{S}$ samples the distribution of the sizes of the future message packets (i.e., $\left\langle (k_{i+1}^{(j)}, \ldots, k_t^{(j)}) \sim D_{k_0, \ldots, k_i} \mid j \in \{0, \ldots, \mathcal{S}-1\} \right\rangle$). We use a similar technique to empirical risk minimization over the $\mathcal{S}$ samples to set $f_i^{(\epsilon)}$ to the value leading to lowest expected

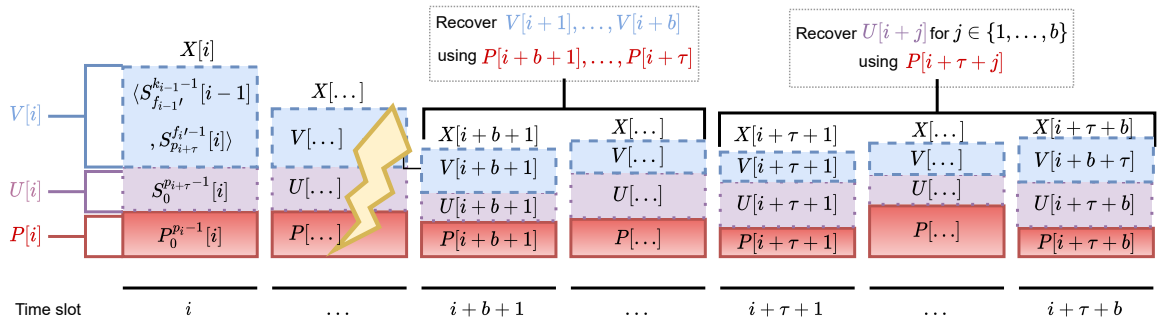

Fig. 4: An example of how the $(\tau, b, t, \langle f_i \mid i \in [t]\rangle)-$Spread Code recovers a burst of length b starting in time slot $(i+1)$.

number of symbols being sent by a rate-optimal offline code. Specifically, for any $i = 0, \ldots, t$, $j \in [\mathcal{S}-1]$ and $l \in [k_i]$, let

$$z_{i,j,l} = \sum_{r=i}^{t} p_r^{(IP)},$$

where $p_i^{(IP)}, \ldots, p_t^{(IP)}$ are variables used by the IP of Algorithm 2 given $X[0], \ldots, X[i-1]$, and $f_i^{(IP)} = l$. Then

$$f_i^{(\epsilon)} = \arg\min_{l\in[k_i]} \frac{1}{\mathcal{S}-1} \sum_{j\in[\mathcal{S}-1]} z_{i,j,l}. \tag{12}$$

The key observation to interpret the choice of $f_i^{(\epsilon)}$ is that the number of parity symbols sent corresponding to message packet $S[i]$, namely $p_{i+\tau}$, is monotonically non-decreasing as $f_i^{(\epsilon)}$ increases. Smaller values of $f_i^{(\epsilon)}$ lead to smaller values of $p_{i+\tau}$ to exploit the parity symbols sent before time slot $(i+\tau)$. This strategy is effective when the next several message packets are likely small. Thus, a small $P[i+\tau]$ ensures that the next several message packets are recovered when some of $X[i+2], \ldots, X[i+\tau-1]$ are lost. In contrast, larger values of $f_i^{(\epsilon)}$ promote larger values of $p_{i+\tau}$, which is suitable when the next several message packets are likely to be large. A large $P[i+\tau]$ is still useful even if a burst starts after receiving $S[i]$.

To show that the $(\tau, b, t)-$Spread ML Code is approximately rate optimal, we analyze the number of extra symbols it sends compared to an optimal scheme as follows.

Definition 3 (Regret): The *regret*, $\mathcal{R}_{k_i,\ldots,k_t}\left(f_i^{(\epsilon)}\right)$, for the message packet size sequence $k_0, \ldots, k_t$ is the number of extra symbols sent under Algorithm 2 when $f_i^{(\epsilon)}$ is used instead of the best offline policy, f_i'.

The number of extra symbols sent compared to an optimal offline scheme is $\sum_{i=0}^{t} \mathcal{R}_{k_i,\ldots,k_t}\left(f_i^{(\epsilon)}\right)$, as is verified in [1].

Next, we bound the expected regret of the $(\tau, b, t)-$Spread ML Code from spreading message symbols for any time slot.

Lemma 4: For any (τ, b, t), $\mathcal{S} \geq \sqrt{\ln(\frac{8m^2}{\epsilon})}\frac{2\sqrt{2}m^3}{\epsilon}$ samples from the side information, where m is the maximum size of a message packet, $i \in [t]$, $k_0, \ldots, k_{i-1}$, and $f_i' \in [k_i]$,

$$\mathrm{E}_{k_{i+1},\ldots,k_t}\left[\mathcal{R}_{k_i,\ldots,k_t}\left(f_i^{(\epsilon)}\right) - \mathcal{R}_{k_i,\ldots,k_t}\left(f_i'\right)\right] \leq \epsilon. \tag{13}$$

Proof sketch: If $k_i = 0$, $f_i = f_i^{(\epsilon)}$. Otherwise, replicating $f_i^{(\epsilon)}$ symbols ensures $\mathcal{R}_{k_i,\ldots,k_t}\left(f_i^{(\epsilon)}\right) \leq m$. Consequently,

$$\mathrm{E}_{k_{i+1},\ldots,k_t}\left[\mathcal{R}_{k_i,\ldots,k_t}\left(f_i^{(\epsilon)}\right)\right] \leq m$$

$$\mathrm{Var}_{k_{i+1},\ldots,k_t}\left(\mathcal{R}_{k_i,\ldots,k_t}\left(f_i^{(\epsilon)}\right)\right) \leq m^2.$$

At a high level, the empirical mean over the $\mathcal{S}$ samples from the side information accurately approximates the expected regret for each policy (Hoeffding bound [50]). The best policy over the samples is suitable. A full proof is shown in [1]. ■

Finally, we show that the expected online-optimal-rate, denoted as "$R^{(\mathrm{E},Opt)}$," is within ϵ of the expected rate of the $(\tau, b, t)-$Spread ML Code, denoted as

$$R^{(\mathrm{E})} = \mathrm{E}_{k_0,\ldots,k_t}\left[\frac{\sum_{i=0}^{t} k_i}{\sum_{i=0}^{t} k_i + p_i}\right]. \tag{14}$$

Theorem 2: For any (τ, b, t) and for $\mathcal{S} \geq \sqrt{\ln(\frac{8m^2}{\epsilon})}\frac{2\sqrt{2}m^3}{\epsilon}$ samples from the side information, where m is the maximum size of a message packet, $(R^{(\mathrm{E},Opt)} - R^{(\mathrm{E})}) < \epsilon$.

Proof sketch: By Lemma 3, there exists some online $(\tau, b, t, \langle f_i' \mid i \in [t]\rangle)-$Spread Code with optimal expected rate. In expectation, the $(\tau, b, t)-$Spread ML Code sends at most

$$\sum_{i=0}^{t} \mathcal{R}_{k_i,\ldots,k_t}\left(f_i^{(\epsilon)}\right) - \mathcal{R}_{k_i,\ldots,k_t}\left(f_i'\right) \leq \epsilon \sum_{i=0}^{t} \mathbb{1}[k_i > 0],$$

more symbols than the optimal code does, leading to $(R^{(\mathrm{E},Opt)} - R^{(\mathrm{E})}) < \epsilon$. A full proof is shown in [1] ■

More generally, any criteria for spreading message symbols with a sufficiently small expected regret is suitable.

Corollary 2: Theorem 2 holds when any criteria for spreading message symbols is substituted for Eq. (12) if the criteria satisfies Eq. (13) for all $k_0, \ldots, k_t$, $i \in [t]$, and $f_i' \in [k_i]$.

VI. Conclusion

Inspired by the growing field of learning-augmented algorithms, this work introduces a new methodology for constructing online streaming codes that combines machine learning with algebraic coding theory tools. The approach is to (a) isolate the component that can benefit from machine learning, (b) solve the offline version of the problem by integrating optimization with algebraic coding theory techniques, and (c) convert the offline scheme into an online one using a learning-based approach. This strategy is applicable beyond the setting considered in this paper, including numerous other settings for real-time streaming communication.

Acknowledgment

This work was funded in part by NSF grant CCF-1910813.

REFERENCES

[1] M. Rudow and K. V. Rashmi, "Learning-based streaming codes are approximately optimal for variable-size messages," *arXiv preprint arXiv:2205.08521*, 2022.

[2] A. Badr, A. Khisti, W. Tan, and J. Apostolopoulos, "Perfecting protection for interactive multimedia: A survey of forward error correction for low-delay interactive applications," *IEEE Signal Processing Magazine*, vol. 34, no. 2, pp. 95–113, March 2017.

[3] E. Martinian and C. . W. Sundberg, "Burst erasure correction codes with low decoding delay," *IEEE Trans. Inf. Theory*, vol. 50, no. 10, pp. 2494–2502, Oct 2004.

[4] E. Martinian and M. Trott, "Delay-optimal burst erasure code construction," in *ISIT*, June 2007, pp. 1006–1010.

[5] A. Badr, P. Patil, A. Khisti, W. Tan, and J. Apostolopoulos, "Layered constructions for low-delay streaming codes," *IEEE Trans. Inf. Theory*, vol. 63, no. 1, pp. 111–141, Jan 2017.

[6] S. L. Fong, A. Khisti, B. Li, W. Tan, X. Zhu, and J. Apostolopoulos, "Optimal streaming codes for channels with burst and arbitrary erasures," *IEEE Trans. Inf. Theory*, vol. 65, no. 7, pp. 4274–4292, July 2019.

[7] M. N. Krishnan and P. V. Kumar, "Rate-optimal streaming codes for channels with burst and isolated erasures," in *ISIT*, June 2018, pp. 1809–1813.

[8] M. N. Krishnan, D. Shukla, and P. V. Kumar, "Rate-optimal streaming codes for channels with burst and random erasures," *IEEE Trans. Inf. Theory*, vol. 66, no. 8, pp. 4869–4891, 2020.

[9] E. Domanovitz, S. L. Fong, and A. Khisti, "An explicit rate-optimal streaming code for channels with burst and arbitrary erasures," *IEEE Transactions on Information Theory*, vol. 68, no. 1, pp. 47–65, 2022.

[10] A. Badr, A. Khisti, and E. Martinian, "Diversity embedded streaming erasure codes (de-sco): Constructions and optimality," *IEEE J. Sel. Areas Inf. Theory*, vol. 29, no. 5, pp. 1042–1054, May 2011.

[11] N. Adler and Y. Cassuto, "Burst-erasure correcting codes with optimal average delay," *IEEE Trans. Inf. Theory*, vol. 63, no. 5, pp. 2848–2865, May 2017.

[12] D. Leong and T. Ho, "Erasure coding for real-time streaming," in *ISIT*, July 2012, pp. 289–293.

[13] D. Leong, A. Qureshi, and T. Ho, "On coding for real-time streaming under packet erasures," in *ISIT*, July 2013, pp. 1012–1016.

[14] S. L. Fong, A. Khisti, B. Li, W.-T. Tan, X. Zhu, and J. Apostolopoulos, "Optimal streaming erasure codes over the three-node relay network," *IEEE Trans. Inf. Theory*, vol. 66, no. 5, pp. 2696–2712, 2020.

[15] A. Badr, A. Khisti, W.-t. Tan, X. Zhu, and J. Apostolopoulos, "Fec for voip using dual-delay streaming codes," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, 2017, pp. 1–9.

[16] Z. Li, A. Khisti, and B. Girod, "Correcting erasure bursts with minimum decoding delay," in *Conf. Rec. Asilomar Conf. Signals Syst. Comput.*, Nov 2011, pp. 33–39.

[17] Y. Wei and T. Ho, "On prioritized coding for real-time streaming under packet erasures," in *Allerton*. IEEE, 2013, pp. 327–334.

[18] P.-W. Su, Y.-C. Huang, S.-C. Lin, I.-H. Wang, and C.-C. Wang, "Random linear streaming codes in the finite memory length and decoding deadline regime," in *ISIT*, 2021, pp. 730–735.

[19] M. Rudow and K. Rashmi, "Streaming codes for variable-size messages," *IEEE Transactions on Information Theory*, pp. 1–1, 2022.

[20] ——, "Online versus offline rate in streaming codes for variable-size messages," in *ISIT*. IEEE, 2020, pp. 509–514.

[21] T. Lykouris and S. Vassilvtiskii, "Competitive caching with machine learned advice," in *International Conference on Machine Learning*. PMLR, 2018, pp. 3296–3305.

[22] A. Antoniadis, C. Coester, M. Elias, A. Polak, and B. Simon, "Online metric algorithms with untrusted predictions," in *International Conference on Machine Learning*. PMLR, 2020, pp. 345–355.

[23] M. Mitzenmacher, "A model for learned bloom filters and optimizing by sandwiching," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[24] T. Kraska, A. Beutel, E. H. Chi, J. Dean, and N. Polyzotis, "The case for learned index structures," in *Proceedings of the 2018 international conference on management of data*, 2018, pp. 489–504.

[25] S. Lattanzi, T. Lavastida, B. Moseley, and S. Vassilvitskii, "Online scheduling via learned weights," in *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2020, pp. 1859–1877.

[26] É. Bamas, A. Maggiori, and O. Svensson, "The primal-dual method for learning augmented algorithms," *Advances in Neural Information Processing Systems*, vol. 33, pp. 20 083–20 094, 2020.

[27] M. Mitzenmacher and S. Vassilvitskii, "Algorithms with predictions," *arXiv preprint arXiv:2006.09123*, 2020.

[28] C.-Y. Hsu, P. Indyk, D. Katabi, and A. Vakilian, "Learning-based frequency estimation algorithms." in *International Conference on Learning Representations*, 2019.

[29] T. Jiang, Y. Li, H. Lin, Y. Ruan, and D. P. Woodruff, "Learning-augmented data stream algorithms," in *International Conference on Learning Representations*, 2019.

[30] K. Anand, R. Ge, and D. Panigrahi, "Customizing ml predictions for online algorithms," in *International Conference on Machine Learning*. PMLR, 2020, pp. 303–313.

[31] T. Gruber, S. Cammerer, J. Hoydis, and S. t. Brink, "On deep learning-based channel decoding," in *2017 51st Annual Conference on Information Sciences and Systems (CISS)*, 2017, pp. 1–6.

[32] E. Nachmani, E. Marciano, L. Lugosch, W. J. Gross, D. Burshtein, and Y. Be'ery, "Deep learning methods for improved decoding of linear codes," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 119–131, 2018.

[33] L. Lugosch and W. J. Gross, "Neural offset min-sum decoding," in *ISIT*, 2017, pp. 1361–1365.

[34] S. Dörner, S. Cammerer, J. Hoydis, and S. t. Brink, "Deep learning based communication over the air," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 132–143, 2018.

[35] S. Cammerer, T. Gruber, J. Hoydis, and S. ten Brink, "Scaling deep learning-based decoding of polar codes via partitioning," in *GLOBECOM 2017 - 2017 IEEE Global Commun. Conference*, 2017, pp. 1–6.

[36] Y. Jiang, H. Kim, H. Asnani, S. Kannan, S. Oh, and P. Viswanath, "Turbo autoencoder: Deep learning based channel codes for point-to-point communication channels," *Advances in neural information processing systems*, vol. 32, pp. 2758–2768, 2019.

[37] Y. Jiang, S. Kannan, H. Kim, S. Oh, H. Asnani, and P. Viswanath, "Deepturbo: Deep turbo decoder," in *SPAWC*. IEEE, 2019, pp. 1–5.

[38] E. Nachmani and L. Wolf, "Hyper-graph-network decoders for block codes," *Advances in Neural Information Processing Systems*, vol. 32, pp. 2329–2339, 2019.

[39] A. V. Makkuva, X. Liu, M. V. Jamali, H. Mahdavifar, S. Oh, and P. Viswanath, "Ko codes: inventing nonlinear encoding and decoding for reliable wireless communication via deep-learning," in *International Conference on Machine Learning*. PMLR, 2021, pp. 7368–7378.

[40] H. Kim, Y. Jiang, S. Kannan, S. Oh, and P. Viswanath, "Deepcode: Feedback codes via deep learning," *Advances in neural information processing systems*, vol. 31, 2018.

[41] D. B. Kurka and D. Gündüz, "Deepjscc-f: Deep joint source-channel coding of images with feedback," *IEEE J. Sel. Areas Inf. Theory*, vol. 1, no. 1, pp. 178–193, 2020.

[42] J. Kosaian, K. V. Rashmi, and S. Venkataraman, "Learning-based coded computation," *IEEE J. Sel. Areas Inf. Theory*, vol. 1, no. 1, pp. 227–236, 2020.

[43] H. V. Krishna Giri Narra, Z. Lin, G. Ananthanarayanan, S. Avestimehr, and M. Annavaram, "Collage inference: Using coded redundancy for lowering latency variation in distributed image classification systems," in *ICDCS*, 2020, pp. 453–463.

[44] Y.-S. Jeon, S.-N. Hong, and N. Lee, "Blind detection for mimo systems with low-resolution adcs using supervised learning," in *2017 IEEE International Conference on Communications (ICC)*, 2017, pp. 1–6.

[45] N. Samuel, T. Diskin, and A. Wiesel, "Learning to detect," *EEE Trans. Signal Process.*, vol. 67, no. 10, pp. 2554–2564, 2019.

[46] N. Farsad and A. Goldsmith, "Detection algorithms for communication systems using deep learning," 2017.

[47] G. Gui, H. Huang, Y. Song, and H. Sari, "Deep learning for an effective nonorthogonal multiple access scheme," *IEEE Trans. Veh. Technol.*, vol. 67, no. 9, pp. 8440–8450, 2018.

[48] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Trans. Cogn. Commun. Netw.*, vol. 3, no. 4, pp. 563–575, 2017.

[49] T. J. O'Shea, K. Karra, and T. C. Clancy, "Learning to communicate: Channel auto-encoders, domain specific regularizers, and attention," in *2016 IEEE Int. Symp. Signal Process. Inf. Technol.*, 2016, pp. 223–228.

[50] W. Hoeffding, "Probability inequalities for sums of bounded random variables," in *The collected works of Wassily Hoeffding*. Springer, 1994, pp. 409–426.