Reordered Elias Gamma Error Correction Codes for the Near-Capacity Transmission of Multimedia Information

Tao Wang, Matthew F. Brejza, Wenbo Zhang, Robert G. Maunder and Lajos Hanzo
School of ECS, University of Southampton, SO17 1BJ, United Kingdom.
Tel: +44-23-8059 3125, Fax: +44-23-8059 4508
Email: {tw08r,mfb2g09,wz4g11,rm,lh}@ecs.soton.ac.uk, http://www-mobile.ecs.soton.ac.uk

***Abstract*—A novel Joint Source and Channel Coding (JSCC) scheme is proposed, which we refer to as the *Reordered Elias Gamma Error Correction* (REGEC) code. Like the recently-proposed Unary Error Correction (UEC) code and Elias Gamma Error Correction (EGEC) code, the proposed code facilitates the practical near-capacity transmission of source symbol values that are randomly selected from a large or infinite alphabet. However, in contrast to the UEC code, both the EGEC and our proposed REGEC codes are universal codes, facilitating the transmission of source symbol values that are randomly selected using any monotonic probability distribution. However, the EGEC code has a complicated structure comprising two parts, where unequal error protection is required to balance the two parts with the aid of a specific parametrization that must be tailored to the source distribution, preventing its employment for unknown or non-stationary sources. By contrast, the proposed REGEC code does not need unequal error protection, hence its parametrization does not have to be tailored to the particular source distribution, and thus the REGEC code is a more attractive scheme. More explicitly, our REGEC code has a simple structure comprising only a single part, which does not suffer from the delay and loss of synchronization that are associated with the two parts of the EGEC code. In a particular practical scenario, where the source symbols obey a specific Zeta probability distribution, our REGEC scheme is shown to offer gains of up to 0.9 dB over the best of JSCC and Separate Source and Channel Coding (SSCC) benchmarkers, when QPSK modulation is employed for transmission over an uncorrelated narrowband Rayleigh fading channel. In the scenario where the source symbols obey the distribution produced by the H.265 video codec, our REGEC scheme is shown to offer a gain of 0.7 dB over the SSCC benchmarker. These gains are achieved for free, without increasing the required transmit-duration, transmit-bandwidth, transmit-energy or decoding complexity.**

***Index Terms*—Source coding, Video coding, Channel coding, Channel capacity, Iterative decoding**

List of Acronyms

ACS	Add, Compare and Select
BCJR	Bahl-Cocke-Jelinek-Raviv
CC	Convolutional Code
DCMC	Discrete-input Continuous-output Memoryless Channel
EEP	Equal Error Protection
EG	Elias Gamma
EGEC	Elias Gamma Error Correction
EXIT	EXtrinsic Information Transfer
ExpG	Exponential Golomb
FD	Free Distance
FLC	Fixed Length Code
HD	Hamming Distance
IID	Independent and Identically Distributed
JSCC	Joint Source and Channel Code
LDPC	Low Density Parity Check
LLR	Logarithmic Likelihood Ratio
MI	Mutual Information
ML	Maximum Likelihood
QPSK	Quaternary Phase Shift Keying
REG	Reordered Elias Gamma
REGEC	Reordered Elias Gamma Error Correction
RV	Random Variable
SER	Symbol Error Ratio
SNR	Signal to Noise Ratio
SSCC	Separate Source and Channel Coding
UEC	Unary Error Correction
UEP	Unequal Error Protection
URC	Unity Rate Code
VLEC	Variable Length Error Correction

List of Symbols

a	Number of symbols.
b	Number of bits.
$\mathbb{C}$	Trellis codewords set.
c	Trellis codeword.
$\mathbf{D}$	Vector of symbol Random Variables (RVs).
$\mathbf{d}$	Symbol vector.
d_f	The free distance of an error correction code.
f	Number of unary states.
H	Entropy.
h	Number of Fixed Length Code (FLC) states.
L	Symbol value limit for a finite-cardinality source set.
l	The average codeword length.
$\mathbf{m}$	Trellis path.
m_j	Trellis state.

The financial support of the EPSRC, Swindon UK under the grants EP/J015520/1 and EP/L010550/1 of the European Research Council's Advanced Fellow Grant Beam-me-up, as well as that of the TSB, Swindon UK under the auspices of grant TS/L009390/1 is gratefully acknowledged. The research data for this paper is available at http://dx.doi.org/10.5258/SOTON/398289

n	The number of bits in each trellis codeword.
$P(\cdot)$	The probability function.
p_1	Probability of occurrence of symbols having the value.
R	Coding rate.
r	Number of trellis states.
$\mathbf{t}$	Sub-symbol vector for EGEC(FLC-CC) encoder.
$\mathbf{T}$	Vector of sub-symbol RVs for EGEC(FLC-CC) encoder
$\mathbf{u}$	Binarization of sub-symbol vector $\mathbf{t}$.
$\mathbf{v}$	Interleaved bit vector of binary vector $\mathbf{u}$.
$\mathbf{w}$	The encoded output bit vector for EGEC(FLC) encoder.
$\mathbf{x}$	Sub-symbol vector for EGEC(UEC) encoder.
$\mathbf{X}$	Vector of sub-symbol RVs for EGEC(UEC) encoder.
$\mathbf{y}$	Binarization of sub-symbol vector $\mathbf{x}$.
$\mathbf{z}$	The encoded output bit vector for EGEC(UEC) encoder.
$\tilde{(\cdot)}^a$	The *a priori* Logarithmic Likelihood Ratio (LLR) vector pertaining to the corresponding symbol/bit vector .
$\tilde{(\cdot)}^e$	The *extrinsic* LLR vector pertaining to the corresponding symbol/bit vector.
$\tilde{(\cdot)}^p$	The *a posteriori* LLRs pertaining to the symbol/bit vector.
$\hat{(\cdot)}$	Reconstruction of the corresponding symbol or bit vector.

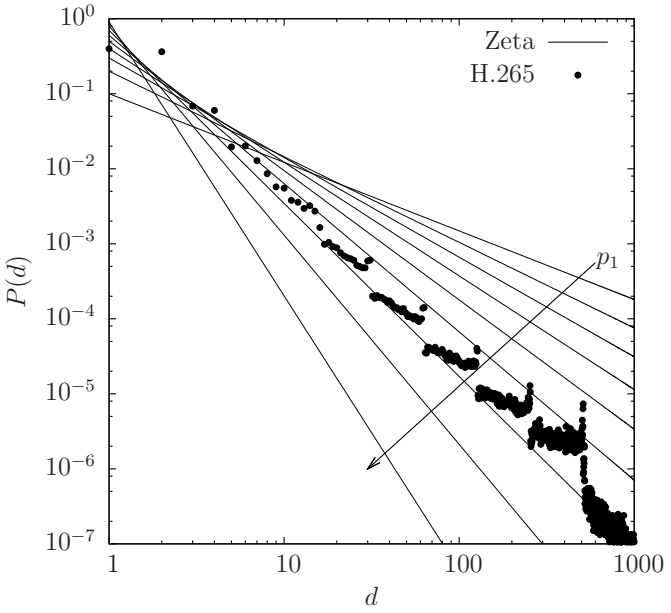


Fig. 1. The finite Zeta-like probability distribution for $L = 1000$ and $p_1 \in \{0.1, 0.2, \ldots, 0.7, 0.8, 0.9\}$ which have the corresponding entropies of $H_D \in \{8.0297, 6.5140, 5.2299, 4.1314, 3.1974, 2.4085, 1.7388, 1.1541, 0.6118\}$, as well as the H.265 distribution, which has the entropy $H_D = 2.3922$. This was obtained by recording the values of the 112.9 million symbols when the HM-9.0 H.265 video encoder employs the 'encoder_lowdelay_main.cfg' and 'encoder_random_access_main.cfg' configurations to encode the 220 s of video that are comprised by 4:2:0 versions of the 24 video test sequences that are commonly used for testing in H.265 [7, page 94].

I. Introduction

Multimedia codecs such as H.264 [1] and H.265 [2] typically produce symbols that have a wide range of values. Our previous work [3, Figure 1] and Figure 1 demonstrate that both H.264 and H.265 produce symbol values that may be represented using positive integers having values of up to around 1000, where higher values are observed with lower probabilities. This is characteristic of Zipf's law [4] and so the symbol values may be modeled using a Zeta probability distribution [4]. These multimedia codecs employ source codes such as the unary code [5] and the Elias Gamma (EG) code [6] for the entropy coding of these symbols. Here, each symbol value is mapped to a different binary codeword, having a variety of different lengths.

In order to facilitate the reliable transmission of multimedia signals over error-prone channels, both source and channel coding is required. The timeline of their development is characterized at a glance in Figures I and I, respectively. Shannon's source-coding and channel-coding separation theorem [8] states that near-capacity communication is theoretically possible, when employing Separate Source and Channel Coding (SSCC). For example, this may be achieved by combining a near-entropy source code, such as an adaptive arithmetic code [9] or a Lempel-Ziv code [10], with a near-capacity channel code, such as a Low Density Parity Check (LDPC) code [11] or a turbo code [12]. However, the source-coding and channel-coding separation theorem relies upon a number of idealized simplifying assumptions, such as having an infinite block-length and random channel errors encountered for transmission over Gaussian channels. Hence this profound theorem has a limited validity for practical finite-delay, finite-complexity schemes communicating over fading channels exhibiting bursty - rather than random - error distributions [13]. Furthermore, near-entropy adaptive arithmetic coding or Lempel-Ziv coding requires both the transmitter and receiver to accurately estimate the occurrence probability of every source symbol. However, the occurrence probability of rare symbol values cannot be accurately estimated until a sufficiently high number of symbols have been generated, imposing an excessive latency which cannot be tolerated in many practical applications. This problem becomes particularly severe, when the symbol values are selected from a set having an infinite cardinality, such as the set of all positive integers. Furthermore, transmission errors may result in corrupting longer codewords in a specific way, where the corrupted codewords mimic a shorter legitimate codeword. This inevitably leads to the loss of synchronization between the transmitter and receiver, potentially causing an avalanche-like propagation of decoding errors.

It is these issues that motivate the employment of structured source codes, such as the unary [5] and EG code [6] in many practical multimedia communication schemes. More specifically, structured source codes operate on the basis of codewords that conform to a particular structure, rather than having a design that is tailored to the specific probabilities of

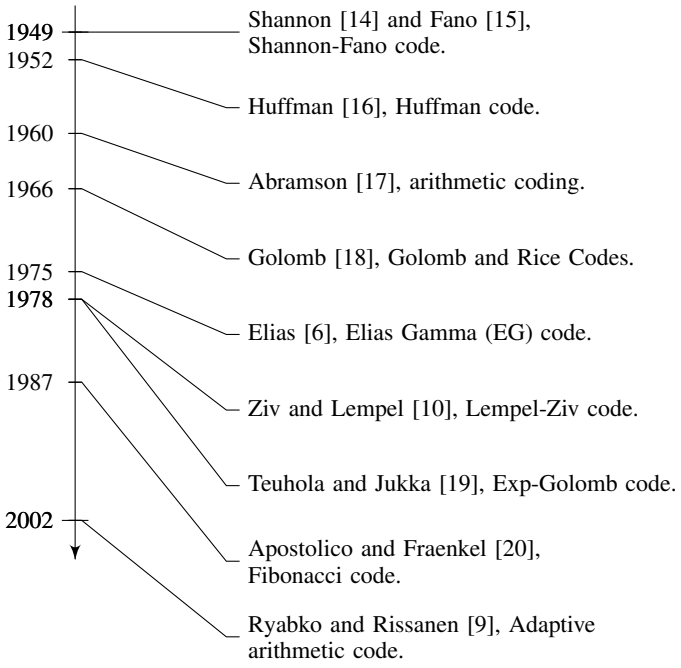


Fig. 2. Timeline of source coding milestones.

occurrence of the symbols generated by the particular source, like arithmetic and Lempel-Ziv codes. Owing to this, structured source codes facilitate the communication of symbols selected from infinite sets, without requiring any knowledge of the corresponding occurrence probabilities at either the transmitter or receiver. Other examples of structured source codes include the Elias delta code [6], the Elias omega code [6], the Even-Rodeh code [21], the Stout code [22] and the Fibonacci code [23], just to name a few. Furthermore, the Exponential Golomb (ExpG) code [19] is a parametrized structured source code, which subsumes the EG code as a special case. As we mentioned before, structured source codes are typically employed in multimedia codecs, where they are invoked for encoding the values of various symbols, such as motion vectors of a sophisticated video codec. However, typically some residual redundancy remains in the source-coded bit-stream when structured source codes are employed for representing symbols that are produced by multimedia codes, hence imposing a capacity loss and preventing near-capacity operation when SSCC is employed [3]. Furthermore, SSCC is sensitive to transmission errors, with a single bit error potentially causing the corruption of several video frames in H.264, for example.

As a remedy, Joint Source and Channel Codes (JSCCs) [42] have been proposed for exploiting the residual redundancy associated with structured source codes, hence avoiding capacity loss. We previously proposed a pair of JSCCs schemes for the near-capacity transmission of source symbols that are randomly selected from a large alphabet, namely the Unary Error Correction (UEC) code [3] and the Elias Gamma Error Correction (EGEC) code [43]. More specifically, our previously proposed UEC code [3] was the first JSCC that

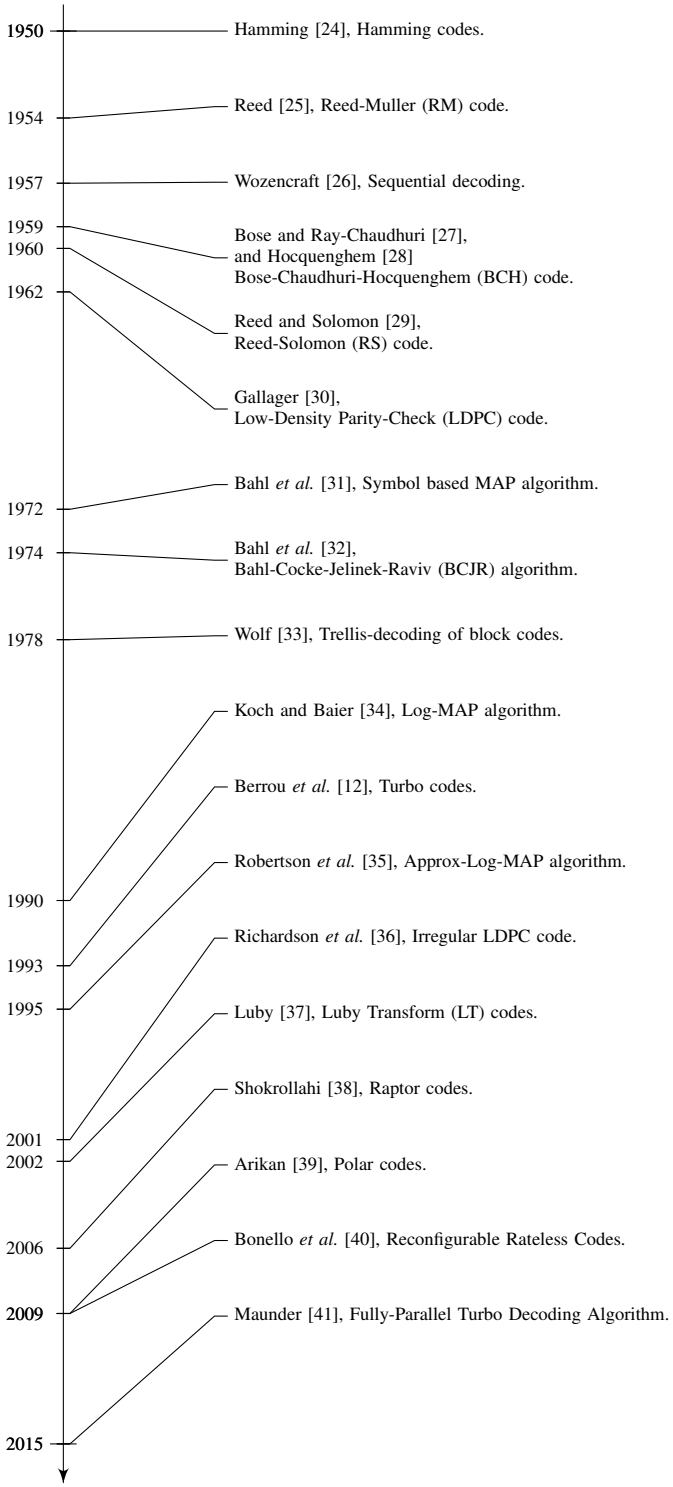


Fig. 3. Timeline of channel coding milestones.

has a low decoding complexity, when invoked for representing symbols values that are selected from an alphabet having a large or infinite cardinality. However, the UEC code has limited applicability, since it is based on the unary code [5], which is a structured source code, but is not a *universal* code, as shown in Figure 4. More specifically, the UEC code only has a finite average codeword length for particular source distributions, including only a limited subset of the Zeta probability distributions that does not include the Zeta distribution that models the symbols produced by H.265 most closely. Motivated by this, we subsequently proposed the EGEC code [43], which was the first *universal* JSCC. More specifically, since the EGEC code is based on the *universal* EG source code [6], it has a finite average codeword length for any mototonic source distribution, in which lower symbol values have greater occurrence probabilities than higher symbol values. Owing to this, the EGEC code was the first JSCC that facilitates the low-complexity near-capacity transmission of symbol values that are randomly selected from a large or infinite alphabet using a widely applicable range of probability distributions. However the EGEC scheme has a complicated structure, comprising two parts, namely the EGEC(UEC) part and the EGEC(FLC-CC) part, as shown in Figure 5(b). The EGEC(UEC) part operates on the basis of the UEC code of [3] while the EGEC(FLC-CC) part employs a serial concatenation of a FLC with a Convolutional Code (CC) and relies on side information provided by the EGEC(UEC) part. Owing to this specific structure, the EGEC(FLC-CC) part cannot be operated until after the operation of the EGEC(UEC) part has been completed, which will cause additional processing delay. Furthermore, if the side information provided by the EGEC(UEC) part contains any decoding errors, the EGEC(FLC-CC) decoder part will become desynchronized, w.r.t to the encoder, hence inflicting a high number of decoding errors. Depending on the particular source probability distribution, the two parts of the EGEC code typically have different error correction performances, with one or other of the parts becoming the dominant limitation of the overall error correction performance. This deficiency can be solved by using puncturing [43] for involving Unequal Error Protection (UEP) for the two parts, giving them equal error correction performances. However the puncturing will impose some capacity loss and it will also increase the complexity of the system, since the punctured bits still have to be decoded during the decoding process. Furthermore, the above-mentioned UEP must be specifically parametrized for a particular source probability distribution. If the actual source distribution is unknown or it is non-stationary, then it will typically fail to match the distributions, hence causing further capacity loss.

Against this background, this paper proposes a *universal* JSCC scheme, which we refer to as the Reordered Elias Gamma Error Correction (REGEC) code. This has a simple structure, which facilitates the near-capacity transmission of symbol values that are randomly selected from large alphabets using *any* arbitrary monotonic probability distribution at a low complexity. Since it is a universal code, the applicability of the REGEC code is not limited to any particular source symbol distribution like the UEC. Furthermore, since the REGEC code has a simple structure comprising only a single constituent part as shown in Figure 5(a), it does not suffer from the delay, from loss of synchronization, loss of capacity or from the increased complexity of puncturing, that are associated with the EGEC. Furthermore, the REGEC code is an attractive solution, since it does not require UEP that is tailored for a specific source distribution, like the EGEC code. Our REGEC code is based on a novel source code, which we refer to as the Reordered Elias Gamma (REG) code, since it reorders the bits in each of the EG codewords for creating a relatively simple structure. Since this is achieved without changing the length of the codewords, the REG code constitutes a universal code, like the EG code. The proposed REGEC code combines the REG source code with a novel trellis-based channel code. Reordering the bits in the EG codewords allows the REGEC trellis to be designed for ensuring that the transitions between its states are synchronous with the transitions between the consecutive codewords in the REG encoded bit sequence. This allows the residual redundancy in the REG encoded-bit sequence to be exploited for error correction by the REGEC trellis decoder, hence facilitating near-capacity operation.

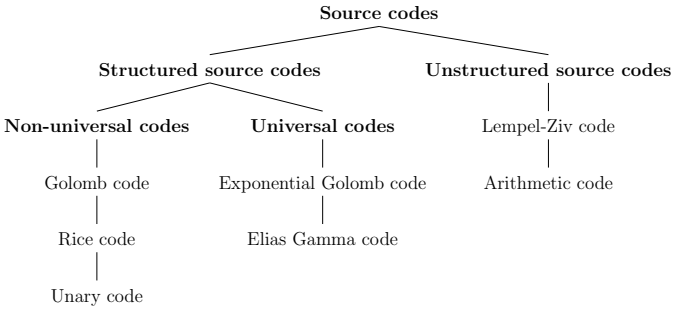


Fig. 4. Taxonomy of selected source codes.

As shown in Figure 6, the rest of this paper is organized as follows. In Section II, we describe the Zeta source probability distribution and generalize the infinite-cardinality source alphabet of our previous work to the case of a finite cardinality, where this cardinality represents an additional parameter to be considered. In Section III, we introduce the novel REG code and describe the structure of the REG codewords. Section IV and V introduce our novel REGEC encoder and decoder, respectively. In Section VI, we analyze the parametrization of the proposed REGEC scheme and demonstrate that it facilitates near-capacity operation. In Section VII, we will consider a wide range of finite Zeta-like probability distributions as well as the H.265 distribution and we will show that our REGEC scheme is capable of offering gains of up to 0.9 dB over the best UEC, EGEC and SSCC benchmarkers in each case, when employing Quaternary Phase Shift Keying (QPSK) for communication over an uncorrelated narrowband Rayleigh fading channel. In the scenario where the source symbols obey H.265 distributions, our REGEC scheme is shown to offer a gain of 0.7 dB over the SSCC benchmarker. Note that these gains are achieved for free, without increasing the required transmit-duration, transmit-bandwidth, transmit-energy or decoding complexity. Finally, we offer our conclusions in

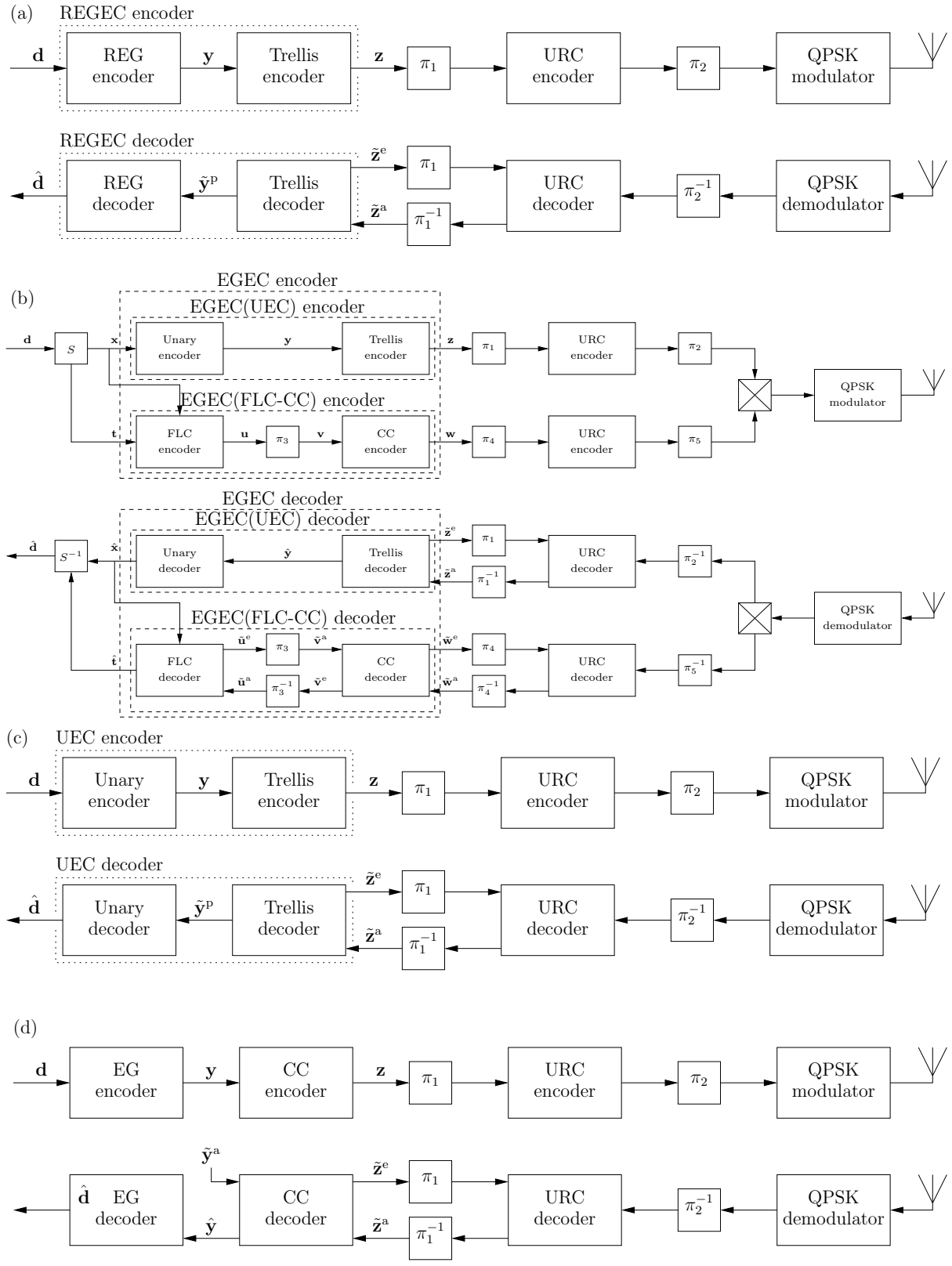


Fig. 5. Schematics of the (a) REGEC, (b) EGEC, (c) UEC JSCC schemes and (d) the EG-CC SSCC scheme, when serially concatenated with URC and Gray-coded QPSK modulation schemes. Bold notation without a diacritic is used to denote a symbol vector or a bit vector. A diacritical hat represents a reconstruction of the symbol or bit vector having the corresponding notation. A diacritical tilde represents an LLR vector pertaining to the bit vector with the corresponding notation. A roman superscript 'a' is employed to denote an *a priori* LLR vector, while 'e' is employed for extrinsic LLR vectors. Furthermore, $\{\pi_1, \ldots, \pi_5\}$ represent interleavers, while $\{\pi_1^{-1}, \ldots, \pi_5^{-1}\}$ represent the corresponding deinterleavers. Puncturing may also be performed in π_2 and π_5, while the corresponding depuncturing operations take place in π_2^{-1} and π_5^{-1}. Multiplexing and demultiplexing is performed in the crossed boxes.

Fig. 6. The structure of the paper

Section VIII.

II. Symbol value sets having a large cardinality

The schemes considered in this paper are designed to convey a vector $\mathbf{d} = [d_i]_{i=1}^{a}$ comprising a symbols. This symbol vector is obtained as the realization of a corresponding vector $\mathbf{D} = [D_i]_{i=1}^{a}$ of Independent and Identically Distributed (IID) RVs. Each RV D_i adopts the symbol value $d \in \mathbb{N}_L$ with probability $\Pr(D_i = d) = P(d)$, where $\mathbb{N}_L = \{1, 2, 3, \dots L\}$ is the finite-cardinality alphabet comprising positive integers with the cardinality L. Our previous contributions [3], [43] characterized the performance of the UEC, EGEC and SSCC schemes invoked for to representing symbols values that are selected from a set having an infinite cardinality. Instead, in this paper we will the use symbol set $\mathbb{N}_L$ having the finite cardinality of $L = 1000$, since the symbol values of H.264 shown in [3, Figure 1] of this treatise and H.265 shown in Figure 1 are selected from an alphabet having a cardinality of approximately 1000. Here, the symbol entropy is given by $H_D = \sum_{d\in\mathbb{N}_L} H[P(d)]$, where $H[p] = p\log_2(1/p)$.

Again Figure 1 exemplifies the distribution of the symbol values that are obtained from the H.265 video encoder, corresponding to a symbol entropy of $H_D = 2.3922$ bits per symbol. Note that these symbol values obey Zipf's law [4], since their distribution may be approximated by the finite Zeta-like distribution. Here, we define the finite Zeta-like distribution as

$$P(d) = \frac{d^{-s}}{H_L^{(s)}}, \tag{1}$$

where $H_L^{(s)}$=$\sum_{d\in\mathbb{N}_L} d^{-s}$ is the generalized harmonic number[1] of order L of s, where $s \in \mathbb{R}$ for finite L. The limit of $L \to \infty$ exists when $s > 1$ and the generalized harmonic number converges to the Riemann Zeta function[2]. The finite Zeta-like distribution may be more conveniently parametrized by the probability of the symbols adapting the most likely value of 1, which is given by $p_1 = \Pr(D_i = 1) = 1/H_L^{(s)}$. In the case of the finite Zeta-like distribution, the symbol entropy is given by

$$H_D = \sum_{d\in\mathbb{N}_L} H[P(d)] = \frac{\ln(H_L^{(s)})}{\ln(2)} - \frac{s(\partial H_L^{(s)}/\partial s)}{\ln(2)H_L^{(s)}}, \tag{2}$$

where $\partial H_L^{(s)}/\partial s = -\sum_{d\in\mathbb{N}_L} \ln(d)d^{-s}$ is the derivative of the harmonic number with respect to s.

III. Reordered Elias Gamma code

As shown in Table I, source encoders such as the unary or EG encoders represent each symbol d_i in the vector $\mathbf{d}$ using a corresponding binary codeword, namely $\mathrm{Unary}(d_i)$ or $\mathrm{EG}(d_i)$, respectively. Note that for the convenience of our ensuing discussions, the unary codewords shown in Table I are the complements of those that are conventionally employed, for example in [3, Table I]. The average codeword length is given by

$$l = \sum_{d\in\mathbb{N}_L} P(d)l(d), \tag{3}$$

where $l(d)$ is the length of the d^{th} codeword.

In the case of a unary code, the length of the codeword $\mathrm{Unary}(d_i)$ is given by $l_{Unary}(d_i) = d_i$, giving an average codeword length of

$$l_{\mathrm{Unary}} = \frac{H_L^{(s-1)}}{H_L^{(s)}}, \tag{4}$$

when the source symbols obey the finite Zeta-like distribution of (1). However, the average unary codeword length l is only finite for $s > 2$ and hence for $p_1 > 0.608$ when L tends to infinity. For the case of the finite Zeta-like distribution having the cardinality $L = 1000$, the average codeword length of the unary code is almost double that of the EG code when $p_1 = 0.608$, as we will characterize below. Despite this, the unary code was used as the basis of the JSCC UEC scheme [3], since its codewords have a relatively simple structure, which can be readily exploited for error correction. More specifically, the structure of the unary codewords can be described by the UEC trellis of [3], without requiring an excessive number of trellis transitions and states.

By contrast, an EG codeword $\mathrm{EG}(d_i)$ has a length of $l_{EG}(d_i) = 2\lfloor\log_2(d_i)\rfloor+1$. When the source symbols obey the

[1]Note the difference between the notation $H_\bullet$ of the entropy and the notation $H_\bullet^{(\bullet)}$ of generalized harmonic number.

[2]When $L \to \infty$, we have $\lim_{L\to\infty} H_L^{(s)} = \zeta(s)$, where $\zeta(s)$=$\sum_{d\in\mathbb{N}} d^{-s}$ is the Riemann Zeta function and $\mathbb{N} = \{1, 2, 3, \dots \infty\}$ is the infinite-cardinality set comprising positive integers.

TABLE I
THE FIRST TWELVE CODEWORDS OF VARIOUS SOURCE CODES

d_i	Unary(d_i)	EG(d_i)	x_i	t_i	Unary(x_i)	FLC(t_i,x_i-1)	REG(d_i)
1	1	1	1	0	1		1
2	01	010	2	0	01	0	001
3	001	011	2	1	01	1	011
4	0001	00100	3	0	001	00	00001
5	00001	00101	3	1	001	01	00011
6	000001	00110	3	2	001	10	01001
7	0000001	00111	3	3	001	11	01011
8	00000001	0001000	4	0	0001	000	0000001
9	000000001	0001001	4	1	0001	001	0000011
10	0000000001	0001010	4	2	0001	010	0001001
11	00000000001	0001011	4	3	0001	011	0001011
12	000000000001	0001100	4	4	0001	100	0100001
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

finite Zeta-like distribution, the average EG codeword length becomes [43]

$$l_{\mathrm{EG}} = 1 - \frac{2(\partial H_L^{(s)}/\partial s)}{\ln(2) H_L^{(s)}} - \frac{2}{H_L^{(s)}} \sum_{d \in \mathbb{N}_L} d^{-s} \mathrm{frac}(\log_2(d)), \quad (5)$$

where the $\mathrm{frac}(\cdot)$ operator yields the fractional part of the operand, where $\mathrm{frac}(3.4) = 0.4$ for example[43]. Note that the average EG codeword length l is finite for all Zeta distributions as $L \to \infty$, not just for those for which have $p_1 > 0.608$. For the case of $L = 1000$, the average EG codeword length is lower than that of the unary code for all cases where $p_1 < 0.794$. However, the conventional EG codewords have a relatively complicated structure, which cannot be readily described by a single trellis and hence cannot be readily exploited for low-complexity error correction using a simple JSCC structure. Owing to this, our previous work[43] was only able to develop a trellis representation of the EG code by decomposing each symbol d_i into two sub-symbols x_i and t_i as shown in Figure 5(b). This was motivated by the observation that each EG codeword $EG(d_i)$ may be considered to be a concatenation of a unary codeword Unary(x_i) and a FLC suffix FLC($t_i, x_i - 1$), where $x_i = \lfloor \log_2(d_i) \rfloor + 1$ and $t_i = d_i - 2^{\lfloor \log_2(d_i) \rfloor}$ as shown in Table I. Here FLC($t_i, x_i - 1$) is the binary representation of the integer t_i using $(x_i - 1)$ bits. As shown in Figure 5(b), each sub-symbol x_i is encoded by the EGEC(UEC) part of the EGEC code, while each sub-symbol t_i is encoded by the EGEC(FLC-CC) part. However, the reliance on these two parts leads to the requirement to tailor the UEP of the two parts for the specific source probability distribution, which may not match with the actual source distribution if it is unknown or non-stationary, as well as imposing for the disadvantages associated with an increased delay, loss of synchronization, capacity loss and increased complexity due to puncturing, as described in Section I.

In order to eliminate the requirement for a complicated code structure comprising two parts, with the design-objective of creating a simple trellis structure, we propose a novel reordering of the bits in each EG codeword. We refer to the reordered code as the REG code, where the generalized structure of each REG codeword is shown in Figure 7. The reordering is conceived as follows. As described above, the

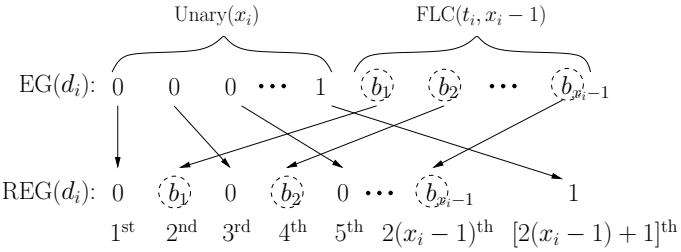


Fig. 7. The reordering of an EG codeword to obtain the corresponding REG codeword

first x_i bits of the conventional EG codeword, EG(d_i) are given by a unary codeword Unary(x_i). These bits become the odd-indexed bits of the our corresponding REG codeword. Notice that the final 1-valued bit in Unary(x_i) becomes the final bit in REG(d_i), since all REG codewords comprise an odd number of bits, in common with all EG codewords. The last $x_i - 1$ bits of the conventional codeword EG(d_i) comprise the FLC codeword FLC($t_i, x_i - 1$), which become the even-indexed bits of the corresponding REG codeword REG(d_i). Since each REG codeword has the same length of $l_{REG(d_i)} = 2\lfloor \log_2(d_i) \rfloor + 1$ as the corresponding EG codeword, the REG code will have the same average codeword length l_{REG} as the EG code, which is given by (5). Therefore, since the EG is a universal code, so too is the REG.

This approach is motivated by the difference in the structures of the unary and EG codewords shown in Table I. In [3], a UEC code was designed for the JSCC of unary-encoded symbols, in order to facilitate near-capacity communication. This is achieved by designing the UEC trellis of [3] for ensuring that the path through the trellis remains synchronized with the unary codewords. More specifically, the UEC trellis uses the logical 1-valued bit at the end of each unary codeword to detect the boundary between consecutive codewords and to trigger a return to a start state. By contrast, maintaining trellis synchronization during the joint source and channel coding of EG-coded symbols is more complicated. This is because the length of the EG codeword depends on the length of its unary prefix, which may be detected using the 1-valued bit at the end. However, an EGEC trellis designed for maintaining synchronization with the EG codewords would require states,

i.e. memory for storing the length of the unary prefix all the way until the end of the FLC suffix is reached, whereupon a return to start state could be triggered. Since the unary prefix can have any length selected from an infinite set, an infinite number of states would be required to store this information, hence preventing the construction of a practical trellis. Instead, we can maintain synchronization by reordering the bits in the EG codeword, so that the logical 1-valued bit at the end of the unary prefix appears instead at the end of the REG codeword. In this way, this logical 1-valued bit may be used for detecting the boundary between consecutive codewords and to trigger a return to start state in the proposed REGEC trellis, which will be introduced in Section IV. In this way, synchronization can be maintained and near-capacity joint source and channel coding can be achieved.

IV. Reordered Elias Gamma Error Correction Encoder

In this section, we introduce the REGEC encoder, which is illustrated in Figure 5(a). In Section IV-A, we discuss the operation of the REG source encoder. The operation of the REGEC trellis is described in Section IV-B. Finally, Section IV-C describes the serial concatenation of the REGEC encoder with the Unity Rate Code (URC) encoder and QPSK modulator of Figure 5(a).

A. Reordered Elias Gamma encoder

The REG encoder of Figure 5(a) represents each symbol d_i in the vector $\mathbf{d}$ using the corresponding REG codeword $\text{REG}(d_i)$, as shown in Table I. These codewords are then concatenated to obtain the b-bit vector $\mathbf{y} = [y_j]_{j=1}^{b}$ shown in Figure 5(a). For example, the vector $\mathbf{x} = [6, 3, 1, 9, 2, 1, 1, 2]$ of $a = 8$ symbols yields the $b = 24$-bit vector $\mathbf{y} = 010010111000001100111001$.

B. Reordered Elias Gamma Error Correction trellis encoder

As shown in Figure 5(a), the bit vector of concatenated REG codewords $\mathbf{y}$ is forwarded to a trellis encoder, which employs a novel REGEC trellis for encoding each bit y_j in the vector $\mathbf{y}$, in order of increasing bit-index j. The trellis comprises b number of concatenated trellis stages of the type depicted in Figure 8. Each trellis stage comprises $2r$ number of transitions between r number of states, where r is required to satisfy $r = 2f + 2$, where f must be even. For example, an $r = 6$-state trellis is shown in Figure 8(a), an $r = 14$-state trellis is shown in Figure 8(b) and the general case is given in Figure 8(c). Each successive bit of y forces the trellis encoder to transition from its particular previous state $m_{j-1} \in \{1, 2, \ldots, r\}$ into a new state $m_j \in \{1, 2, \ldots, r\}$ that is selected from two legitimate alternatives, depending on the bit value y_j. In the trellis stages of Figure 8, $y_j = 0$ forces the trellis to make the dashed transition, while $y_j = 1$ forces the trellis to obey the solid transition. The encoding process always commences from the state $m_0 = 1$. The bit vector $\mathbf{y}$ identifies a path through the trellis, which may be represented by a vector $\mathbf{m} = [m_j]_{j=0}^{b}$ comprising $(b + 1)$ state values. For example, the bit vector $\mathbf{y} = 010010111000001100111001$ yields the path $\mathbf{m} = [1, 3, 6, 4, 6, 1, 3, 6, 1, 2, 4, 6, 4, 6, 4, 5, 2, 4, 6, 1, 2, 1, 3, 5, 2, 1]$ through the $r = 6$-state trellis of Figure 8(a).

As discussed in Section III, the odd-indexed bits in the REG codewords derive from a unary codeword, while the even-indexed bits come from an FLC codeword. These unary and FLC bits force the trellis path into different sub-sets of the r trellis states. More specifically, we decompose the set of r states in to three sub-sets, namely the unary states, the FLC states and the holding states. The trellis is designed for ensuring that each input bit y_j that is provided by a unary bit causes a transition from one of the first f number of states $m_{j-1} \in \{1, 2, \ldots, f\}$, which we refer to as the unary states, where f must be even. The transition enters a next state m_j, according to

$$m_j = \begin{cases} 1 + \text{odd}(m_{j-1}) \\ \qquad \text{if } \mathrm{y_j} = 1 \text{ and } \mathrm{m_{j-1}} \in \{1, 2, \ldots, \mathrm{f}\} \\ m_{j-1} + f \\ \qquad \text{if } \mathrm{y_j} = 0 \text{ and } \mathrm{m_{j-1}} \in \{1, 2, \ldots, \mathrm{f}\} \end{cases}, \tag{6}$$

where $\text{odd}(\cdot)$ yields 1 if its operand is odd or 0 if it is even. Note that since each REG codeword ends with a unary bit having the value $y_j = 1$, the trellis path $\mathbf{m}$ is guaranteed to enter either state $m_j = 1$ or $m_j = 2$ after each codeword. In this way, the transitions between the states of the REGEC trellis are synchronized with the transitions between the REG codewords in the bit vector $\mathbf{y}$. For the same reason, the trellis path $\mathbf{m}$ is guaranteed to terminate in the state $m_b = 1$ or $m_b = 2$ of the end of the encoding process. By contrast, the other unary bits in each REG codeword have the value $y_j = 0$, which cause transitions to one of the next $(f - 2)$ states $m_{j-1} \in \{f + 1, f + 2, \ldots, 2f - 2\}$, which we refer to as the FLC states, since the next bit will be an FLC bit. This FLC bit is guaranteed to cause a transition from the FLC state to a unary state, since an FLC bit is always followed by a unary bit in the REG codewords. The next state m_j, is selected according to

$$m_j = \begin{cases} m_{j-1} - f + 2 \cdot \text{odd}(m_{j-1}) + 1 \\ \qquad \text{if } \mathrm{y_j} = 1 \text{ and } \mathrm{m_{j-1}} \in \{\mathrm{f} + 1, \ldots, 2\mathrm{f} - 2\} \\ m_{j-1} - f + 2 \\ \qquad \text{if } \mathrm{y_j} = 0 \text{ and } \mathrm{m_{j-1}} \in \{\mathrm{f} + 1, \ldots, 2\mathrm{f} - 2\} \end{cases}. \tag{7}$$

Observe that when $f = 2$, there are no FLC states in the trellis, as shown in Figure 8(a). Note that REG codewords having a length $l(d_i) \leqslant (2f - 2)$ cause the path $\mathbf{m}$ to enter only the unary and FLC states described above. However, REG codewords having a length $l(d_i) > (2f - 2)$ require four additional states, which we refer to as the holding states, since they act as a 'holding pattern' for the bits in the REG codeword from the $(2f - 1)^{\text{st}}$ bit onward. More specifically, the FLC holding states $m_j \in \{2f - 1, 2f\}$ are entered into, if the unary bit $y_j = 0$ is encountered, while being in one of the unary states of the set $m_{j-1} \in \{f - 1, f\}$, as shown in (6). Upon

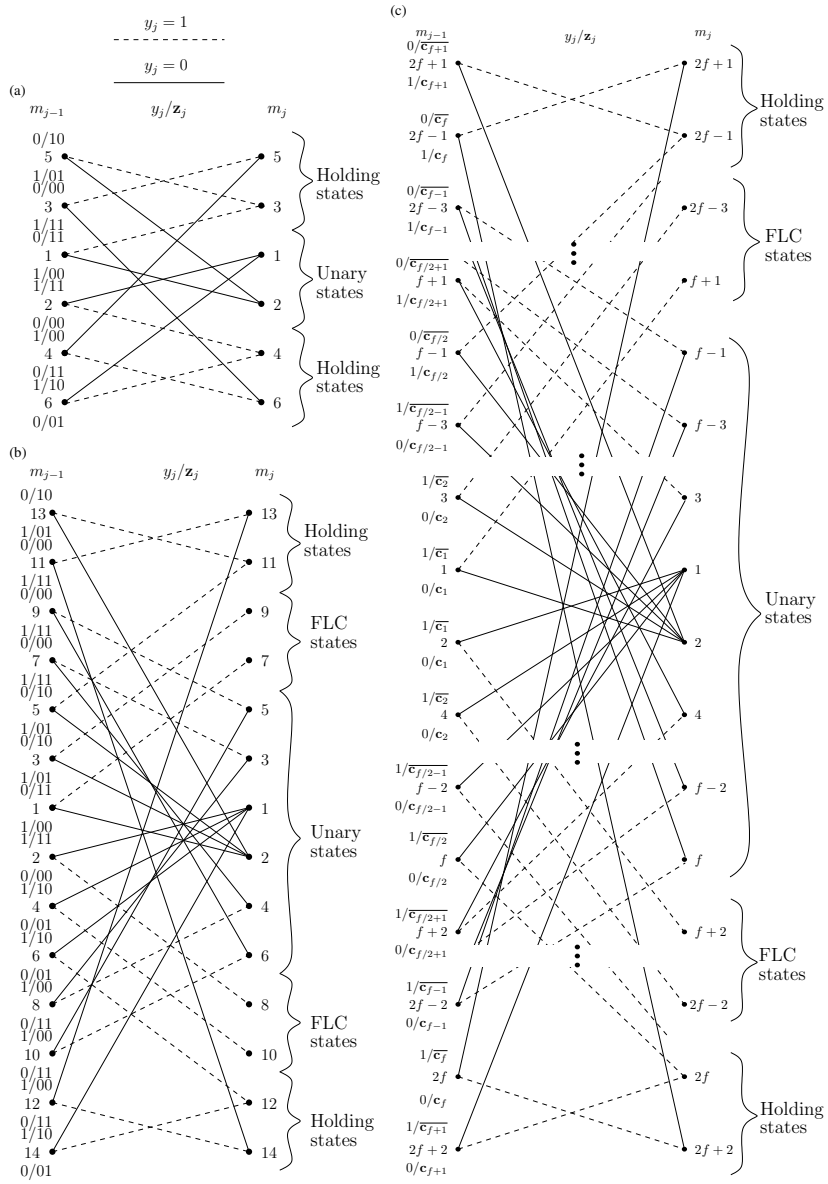


Fig. 8. (a) An $f = 2$, $r = 6$-state $n = 2$-bit REGEC trellis using the codebook $\mathbb{C} = [00; 11; 01]$. (b) An $f = 6$, $r = 14$-state $n = 2$-bit REGEC trellis where using the codebook $\mathbb{C} = [00; 01; 01; 11; 11; 11; 01]$ which is constructed by extending the codebook $\mathbb{C} = [00; 11; 01]$. (c) The generalized REGEC trellis, having r states and n-bit codewords, where $\mathbb{C} = [\mathbf{c}_1; \mathbf{c}_2; \ldots, \mathbf{c}_f, \mathbf{c}_{f+1}]$.

emerging from the FLC holding states $m_{j-1} \in \{2f-1, 2f\}$, the next state will be chosen from the unary holding states of the set $m_j \in \{2f+1, 2f+2\}$ according to

$$m_j = \begin{cases} m_{j-1} + 2 \cdot \text{odd}(m_{j-1}) + 1 \\ \qquad \text{if } \mathrm{y_j} = 1 \text{ and } \mathrm{m_{j-1}} \in \{2\mathrm{f}-1, 2\mathrm{f}\} \\ m_{j-1} + 2 \\ \qquad \text{if } \mathrm{y_j} = 0 \text{ and } \mathrm{m_{j-1}} \in \{2\mathrm{f}-1, 2\mathrm{f}\} \end{cases} . \quad (8)$$

Likewise, upon traversing from the unary holding states $m_{j-1} \in \{2f+1, 2f+2\}$, the next state will be chosen according to

$$m_j = \begin{cases} 1 + \text{odd}(m_{j-1}) \\ \qquad \text{if } \mathrm{y_j} = 1 \text{ and } \mathrm{m_{j-1}} \in \{2\mathrm{f}+1, 2\mathrm{f}+2\} \\ m_{j-1} - 2 \\ \qquad \text{if } \mathrm{y_j} = 0 \text{ and } \mathrm{m_{j-1}} \in \{2\mathrm{f}+1, 2\mathrm{f}+2\} \end{cases} . \quad (9)$$

Note that the trellis path $\mathbf{m}$ will remain in the holding states, as long as unary bits having the value of $y_j = 0$ are encountered. When the final $y_j = 1$-valued unary bit of the REG codeword is encountered, the trellis path returns to state $m_j = 1$ or $m_j = 2$, ready for the start of the next REG codeword. Finally, combining Equations (6) to (9) yields (10). Note that the total number of states is given by $r = (2f + 2)$.

The path $\mathbf{m}$ may be modeled as a particular realization of a vector $\mathbf{M} = [M_j]_{j=0}^{b}$ comprising $(b+1)$ RVs, which are asso-

$$m_j = \begin{cases} 1 + \text{odd}(m_{j-1}) & \text{if } y_j = 1 \text{ and } m_{j-1} \in \{1, 2, \ldots, f, 2f+1, 2f+2\} \\ m_{j-1} + f & \text{if } y_j = 0 \text{ and } m_{j-1} \in \{1, 2, \ldots, f\} \\ m_{j-1} - f + 2 - y_j + 2 \cdot y_j \cdot \text{odd}(m_{j-1}) & \text{if } y_j \in \{0, 1\} \text{ and } m_{j-1} \in \{f+1, f+2, \ldots, 2f-2\} \\ m_{j-1} + y_j \cdot (2 \cdot \text{odd}(m_{j-1}) + 1) + 2 \cdot \text{odd}(y_j + 1) & \text{if } y_j \in \{0, 1\} \text{ and } m_{j-1} \in \{2f-1, 2f\} \\ m_{j-1} - 2 & \text{if } y_j = 0 \text{ and } m_{j-1} \in \{2f+1, 2f+2\} \end{cases} \quad (10)$$

ciated with the transition probabilities $\Pr(M_j = m, M_{j-1} = m') = P(m, m')$ of (11). These transition probabilities depend on the source symbol probabilities $P(d)$, which can be derived by employing the method of [3, Appendix]. In (11), l_1 is the average length of Unary(x_i), as described in Section III. In the case of the finite Zeta-like distribution of (1), l_1 is given by [43]

$$l_1 = 1 - \frac{H_L^{(s)}}{\ln(2) H_L^{(s)}} - \frac{1}{H_L^{(s)}} \sum_{d \in \mathbb{N}_\mathbb{L}} d^{-s} \text{frac}[\log_2(d)]. \quad (12)$$

The conditional transition probabilities $Pr(M_j = m | M_{j-1} = m')$ are given by [3]

$$P(m|m') = \frac{P(m, m')}{\sum_{\check{m}=1}^{r} P(\check{m}, m')}. \quad (13)$$

Once the path $\mathbf{m}$ has been determined, the trellis encoder uses it to represent each bit y_j in the vector $\mathbf{y}$ by an n-bit codeword z_j. This is selected from the matrix of $r/2$ codewords $\mathbb{C} = [\mathbf{c}_1; \mathbf{c}_2; \ldots; \mathbf{c}_{f+1}]$ or from the complementary matrix $\overline{\mathbb{C}} = [\overline{\mathbf{c}_1}; \overline{\mathbf{c}_2}; \ldots; \overline{\mathbf{c}_{f+1}}]$. As shown in Figure 8(c), this is achieved according to

$$\mathbf{z}_j = \begin{cases} \overline{\mathbf{c}_{\lceil m_{j-1}/2 \rceil}} & \text{if } y_j \neq \text{odd}(m_{j-1}) \\ \mathbf{c}_{\lceil m_{j-1}/2 \rceil} & \text{if } y_j = \text{odd}(m_{j-1}) \end{cases}. \quad (14)$$

Following this, the selected codewords are concatenated to obtain the bn-bit vector $\mathbf{z} = [z_k]_{k=1}^{bn}$ of Figure 5. For example, the vector $\mathbf{y} = 010010111000001100111001$ of $b = 24$ bits is represented by the vector $\mathbf{z} = 111101111011111000001101110100010011100011110001$ of $bn = 48$ bits, when employing the $r = 6$-state REGEC trellis of Figure 8(a), with the $n = 2$-bit codebook $\mathbb{C} = [00; 11; 01]$.

Note that the selection of the number of trellis states r is discussed in Section VI-D, while the selection of the codebook $\mathbb{C}$ is discussed in Section VI-E. We emphasize that REGEC trellis encoder operates in a similar manner to a UEC trellis encoder and a CC encoder, but subject to the following important differences, as follows.

1) As in the UEC trellis encoder, a bit having the value of $y_j = 1$ will force a transition from the odd-indexed states at the top half of the REGEC trellis to the even-indexed states in the bottom half and vice-versa. Owing to this symmetry and due to using complementary codewords, the REGEC trellis encoder produces equiprobable bit values for the bit vector $\mathbf{z}$. This results in a bit entropy of $H_z = 1$, which is a necessary condition for avoiding capacity loss, as described in [3]. However, in contract to the unary codewords of the UEC encoder, $y_j = 1$ does not only occur at the end of a REG codeword, resulting in transitions between the top and bottom halves of the REGEC trellis more frequently than only at the end of each codeword. By contrast, CC encoders produce binary values that are not guaranteed to be equiprobable, unless they are specifically parametrized for this purpose, as characterized in [3, Table II].
2) As we described above, the final unary-bit y_j in each REG codeword is guaranteed to induce a transition to either state $m_j = 1$ or state $m_j = 2$ of the REGEC trellis, in analogy with the UEC trellis. However, unlike in the UEC encoder, the particular one from the pair of states $m_b = 1$ or state $m_b = 2$ that is selected at the end of the REGEC trellis path $\mathbf{m}$ depends on more than factors just deciding whether the length a of the symbol vector $\mathbf{d}$ is odd or even. This is due to the transitions between the top and bottom halves of the REGEC trellis that are caused by bits having the value $y_j = 1$ in the middle of REG codewords, as described above. By contrast, in a generalized CC encoder, the trellis path can potentially end in any state, since the transitions between states are not synchronized with the codewords of the source encoder.

Since the binary values in the vector $\mathbf{z}$ are equiprobable, the average coding rate of the REGEC encoder is given by

$$R^{\text{o}} = \frac{H_D}{l_{REG} n}. \quad (15)$$

Here, we employ the roman superscript 'o' to indicate that this coding rate relates to the outer encoder of a serial concatenation, namely the REGEC encoder shown in Figure 5(a).

C. Integration of the REGEC encoder into a transmitter

Following REGEC encoding, the bit vector $\mathbf{z}$ is interleaved by the block π_1, URC encoded [44] and then interleaved again by the block π_2, as shown in Figure 5(a). Puncturing may also be performed within π_2 in order to achieve a particular desired effective throughput η for the transmitter. This is achieved by discarding an appropriate number of bits following interleaving. The inner coding rate R^i is defined by the ratio of bits input into the URC encoder to the number of bits output by π_2, where $R^i > 1$ will be obtained if puncturing is used. Here we employ the roman superscript 'i' to indicate that this coding rate relates to the inner code of a serial concatenation, namely the punctured URC code shown in Figure 5(a). In order to avoid obfuscating the performance analysis of the proposed REGEC scheme by invoking the prevalent high-order modulations schemes routinely used for

$$P(m,m') = \begin{cases} \frac{1}{2l}\left[1 - \sum_{d=1}^{2^{\dot{x}}-1} P(d)\right] & \text{if } m' \in \{1,\ldots,f\} \text{ and } m = m' + f \\ \frac{1}{2l} \sum_{d=2^{\dot{x}-1}}^{2^{\dot{x}}-1} P(d) & \text{if } m' \in \{1,\ldots,f\} \text{ and } m = \text{odd}(m') + 1 \\ \frac{1}{2l} \sum_{x=\ddot{x}+1}^{\lfloor \log_2(L) \rfloor} \sum_{\ddot{t}=0}^{2^{\ddot{x}}-1} \text{odd}(\ddot{t}+1+y_j) \sum_{d=2^x+\ddot{t}\frac{2^x}{2^{\ddot{x}}}}^{2^x+(\ddot{t}+1)\frac{2^x}{2^{\ddot{x}}}-1} P(d) & \text{if } m' \in \{f+1,\ldots,2f-2\} \text{ and } m = \ddot{d} - y_j + 2y_j \cdot \text{odd}(m') \\ \frac{1}{2l} \sum_{x=f}^{\lfloor \log_2(L) \rfloor} \sum_{\mathring{x}=f}^{x} \sum_{\mathring{t}=0}^{2^{\mathring{x}}-1} \text{odd}(\mathring{t}+1+y_j) \sum_{d=2^x+\mathring{t}\frac{2^x}{2^{\mathring{x}}}}^{2^x+(\mathring{t}+1)\frac{2^x}{2^{\mathring{x}}}-1} P(d) & \text{if } m' \in \{2f-1,2f\} \text{ and } m = m' + y_j(2 \cdot \text{odd}(m') + 1) + 2 \cdot \text{odd}(y_j+1) \\ \frac{1}{2l}\left[l_1 - \frac{f}{2} + \sum_{d=1}^{2^{(f/2+1)}-1} p(d)(\lfloor \log_2(d) \rfloor + 1 - \frac{f}{2})\right] & \text{if } m' \in \{2f+1,2f+2\} \text{ and } m = m' - 2 \\ \frac{1}{2l}\left[1 - \sum_{d=1}^{2^{f/2}-1} p(d)\right] & \text{if } m' \in \{2f+1,2f+2\} \text{ and } m = \text{odd}(m') + 1 \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

$$d \in \{1,2...,L\}; \quad y_j \in \{0,1\}; \; \dot{x} = \lceil m'/2 \rceil; \; \ddot{d} = m' - f + 2; \quad \ddot{x} = \lceil \ddot{d}/2 \rceil - 1$$
$$l_1 = P(d_i) \cdot [\lfloor \log_2(d_i) \rfloor + 1]$$

multimedia communication, simple $M = 4$-ary Gray-coded QPSK modulation may be employed for transmission, as shown in Figure 5(a). Note that other mapping schemes or a modulation scheme having a higher modulation-order M can be employed instead, although this may increase the complexity of the receiver, as we will discuss in Section V. The effective throughput of the transmitter is given by

$$\eta = R^o \cdot R^i \cdot \log_2(M). \tag{16}$$

Note that no knowledge of the source probability distribution $P(x)$ is required anywhere in the transmitter.

V. Reordered Elias Gamma Error Correction decoder

In this section, we describe the operation of the REGEC decoder of Figure 5(a). In Section V-A, we discuss the integration of the REGEC decoder with the URC decoder and QPSK demodulator of Figure 5(a). Following this, we detail the operation of the REGEC trellis decoder in Section V-B, while the REG decoder is described in Section V-C.

A. Integration of Reordered Elias Gamma Error Correction decoder into a receiver

In the receiver, soft QPSK demodulation [45], depuncturing and deinterleaving π_2^{-1}, Bahl-Cocke-Jelinek-Raviv (BCJR)-based URC decoding [13] and further deinterleaving π_1^{-1} may be performed, before invoking the proposed REGEC decoder of Figure 5(a). If a higher order modulation scheme were employed, then iterative decoding between the demodulator and the URC decoder would be required to avoid capacity loss, as is the case in any iterative decoding scheme [46]. Note that the receiver is required to employ the same pseudo-random interleaver designs as the transmitter. However, the entire set of interleavers can be generated independently by both the transmitter and receiver using only a single pseudo-random number generator seed. This seed may be hard-coded into both the transmitter and receiver, or may be reliably conveyed using only a very small amount of side information. The REGEC decoder is provided with the $a\ priori$ LLR vector $\tilde{\mathbf{z}}^a$ and in response it generates the extrinsic LLR vector $\tilde{\mathbf{z}}^{\text{e}}$ of Figure 5(a), which may be iteratively exchanged with the serially concatenated URC decoder, until iterative decoding convergence to an infinitesimally low SER is achieved. In turn, the URC decoder may also iteratively exchange extrinsic LLRs with the demodulator [47], in order to avoid capacity loss when a mapping scheme other than Gray coding or when a higher-order modulation scheme is employed. Note that the combination of the URC decoder and the demodulator will have an EXtrinsic Information Transfer (EXIT) curve that reaches the $(1,1)$ point at the top right corner of the EXIT chart[48].

B. Reordered Elias Gamma Error Correction trellis decoder

As shown in Figure 5(a), the REGEC trellis decoder is provided with a vector of *a priori* LLRs $\tilde{\mathbf{z}}^{\text{a}} = [\tilde{z}_k^{\text{a}}]_{k=1}^{bn}$ that pertain to the corresponding bits in the vector $\mathbf{z}$. The trellis decoder applies the BCJR algorithm [32] to a REGEC trellis of the sort shown in Figure 8(c) to consider every legitimate bit vector that could be represented by $\tilde{\mathbf{z}}^a$, having the particular length bn. Here the value of bn is assumed to be perfectly known to the receiver, where the transmitter may employ a small amount of side information to reliably convey this value in practice. Here, the synchronization between the REGEC trellis and the REG codewords is exploited during the BCJR algorithm's γ_t calculation of [32, Equation (9)], by employing the conditional transition probabilities $P(m|m')$ of (11). Note that the REGEC trellis should be terminated at $m_0 = 1$ and at both possibilities for the final state, namely $m_b = 1$ and $m_b = 2$, as described in Section V-A. As shown in Figure 5(a), the BCJR decoder generates the vector of extrinsic LLRs $\tilde{\mathbf{z}}^{\text{e}} = [\tilde{z}_k^{\text{e}}]_{k=1}^{bn}$ which is provided for the next iteration of the concatenated URC decoder's operation. Note that the REGEC trellis decoder's BCJR algorithm has only modest complexity, since it may employ a low number r of states.

Furthermore, it facilitates error correction even if the symbol probability distribution $P(d)$ is unknown, provided that the channel Signal to Noise Ratio (SNR) is sufficiently high, as we shall demonstrate in Section VI-E. In this case, the conditional transition probabilities $P(m|m')$ of (11) will also be unknown and so they are simply omitted from the BCJR algorithm's γ_t calculation.

The transformation of $\tilde{\mathbf{z}}^{\mathrm{a}}$ into $\tilde{\mathbf{z}}^{\mathrm{e}}$ by the trellis decoder of Figure 5(a) may be characterized by plotting the inverted REGEC EXIT curve in an EXIT chart[49], as exemplified in Figure 9. Note that if a suitably designed codebook $\mathbb{C}$

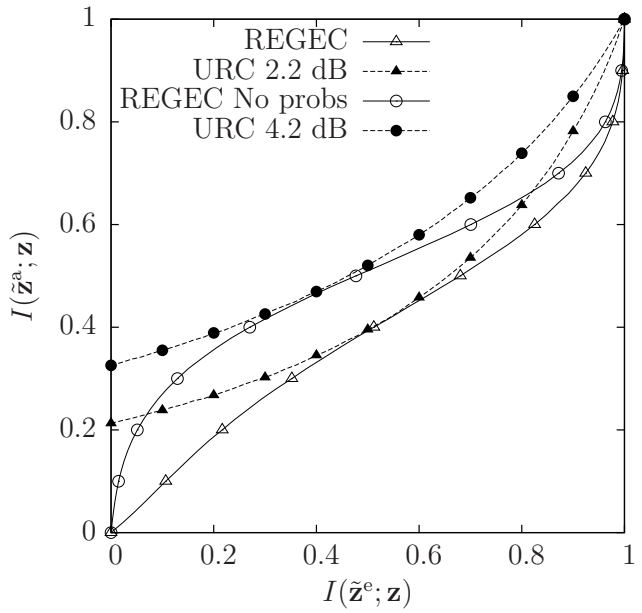


Fig. 9. EXIT charts of the REGEC scheme. Here, the symbols of $\mathbf{d}$ obey a finite Zeta-like distribution having $L = 1000$ and $p_1 = 0.7942$, while the REGEC codewords $\mathbb{C} = [00; 11; 01]$ comprise $n = 2$ bits and result in an REGEC trellis having $r = 6$ states. Furthermore, a URC decoder having $r = 2$ states is concatenated with Gray-coded QPSK modulation, for communication over an uncorrelated narrowband Rayleigh fading channel having various E_{b}/N_0 values.

comprising codewords having at least $n = 2$ bits is employed, then the free distance d_{free} of the REGEC code will be at least two, as it will be quantified in Section VI. In this case the inverted REGEC EXIT curve will reach the $(1, 1)$ point in the top right corner of the EXIT chart [50]. Since the URC decoder and demodulator also have an EXIT curve that reaches the $(1, 1)$ point in the top right corner of the EXIT chart [48] as shown in Figure 9, iterative decoder convergence towards the Maximum Likelihood (ML) performance is facilitated [51].

The EXIT chart area A^o that is situated below the inverted REGEC EXIT curve is given by [43], [52]

$$A^o = \frac{1}{n} \sum_{m'=1}^{r} \sum_{m=1}^{r} P(m, m') \log_2 \left(\frac{1}{P(m|m')} \right). \qquad (17)$$

Note that, the REGEC EXIT chart area A^o is independent of the codebook design, but using different codebooks can affect the shape of the EXIT curve, as will be discussed in Section VI-B. Following the completion of iterative decoding, the REGEC trellis decoder may employ the Viterbi algorithm to generate the vector $\hat{\mathbf{y}} = [\hat{\mathbf{y}}_j]_{j=1}^{b}$ of recovered bits, which pertain to the corresponding bits in the vector $\mathbf{y}$, as shown in Figure 5(a).

C. Reordered Elias Gamma decoder

The decoded bit vector $\hat{\mathbf{y}}$ can be REG decoded in order to obtain the recovered symbol vector $\hat{\mathbf{d}}$ of Figure 5(a). If there are any bit errors in the vector $\hat{\mathbf{y}}$, then we might arrive either at the wrong legitimate REG codeword or fail to find a legitimate codeword. In this case, these bits are discarded. If the decoded symbol vector $\hat{\mathbf{d}}$ does not contain the correct number a of symbols, then an appropriate number of symbols is removed from the end of $\hat{\mathbf{d}}$ or an appropriate number of 1-valued symbols is appended to the end of $\hat{\mathbf{d}}$, accordingly. Here, it is assumed that the REG decoder has perfect knowledge of a. In practice, this value may be fixed in both the transmitter and receiver, or it may be reliably conveyed from transmitter to receiver using a small amount of side information.

VI. Parametrization of the Reordered Elias Gamma Error Correction code

In this section, we discuss the parametrization of the REGEC code. In Section VI-A, we introduce the extension rule of the REGEC codebook extension. In Section VI-B, we analyze the the near-capacity operation of the REGEC decoder. In Section VI-C, we discuss the codebook design of the REGEC trellis encoder, considering the free distance properties of various candidate codebooks. The EXIT curves of the candidate codebooks and their EXIT chart matching are discussed in Section VI-D. Finally we analyze the error floor of the candidate codebooks in Section VI-E and selected a recommended codebook.

A. Reordered Elias Gamma Error Correction codebook extension

As described in Section IV-B, an REGEC trellis having r number of states is parametrized by a set of $r/2$ codewords $\mathbb{C}$, each comprising n number of bits, where $\mathbb{C} = [00; 11; 01]$ in the $r = 6$, $n = 2$ example of Figure 8(a) and $\mathbb{C} = [00; 01; 01; 11; 11; 11; 01]$ in the $r = 14$, $n = 2$ example of Figure 8(b). Any codebook $\mathbb{C}'$ corresponding to a trellis having $r' = 2f' + 2$ number of states can be extended to a codebook $\mathbb{C}''$ corresponding to $r'' > r'$ number of states including several new unary and FLC states. Note that when provided with the same REG-encoded bit vector $\mathbf{y}$, REGEC trellis encoders employing the trellises of Figure 8(a) and Figure 8(b) are guaranteed to generate identical REGEC-encoded bit vectors $\mathbf{z}$, despite using different codebooks $\mathbb{C}$. This is because the $r = 14$ codebook of Figure 8(b) is an *extension* of the $r = 6$ codebook of Figure 8(a). In this way, the use of extension allows a higher number of states r to be used in the REGEC trellis decoder than in the REGEC trellis encoder. This allows us to dynamically change the number of states employed in the decoder in order to strike an attractive trade-off between its performance versus trellis complexity, as characterized in Section VI-B [53].

B. Performance analysis

Near-capacity operation is achieved, when reliable communication can be maintained at transmission throughputs η that approach the Discrete-input Continuous-output Memoryless Channel (DCMC) capacity C [54] that is associated with $M = 4$ QPSK modulation and uncorrelated narrowband Rayleigh fading. This is facilitated, if the following conditions are satisfied [52]:

1) The URC decoder of Figure 5(a) is required to have an EXIT curve having an area beneath it of $A^{\text{i}} = C/[R^{\text{i}} \log_2(M)]$;
2) The area A^{o} beneath the inverted EXIT curve of the REGEC trellis decoder is required to approach the REGEC coding rate R^{o}.

If these two conditions are satisfied, then near-capacity operation will be achieved, when the shape of URC decoder's EXIT curve is closely matched to that of the inverted REGEC EXIT curve. This creates a narrow, but marginally open EXIT chart tunnel, which facilitates iterative decoding convergence towards the ML performance [51].

The first condition listed above is satisfied by a punctured URC code, as discussed in [52]. Figure 10 shows that

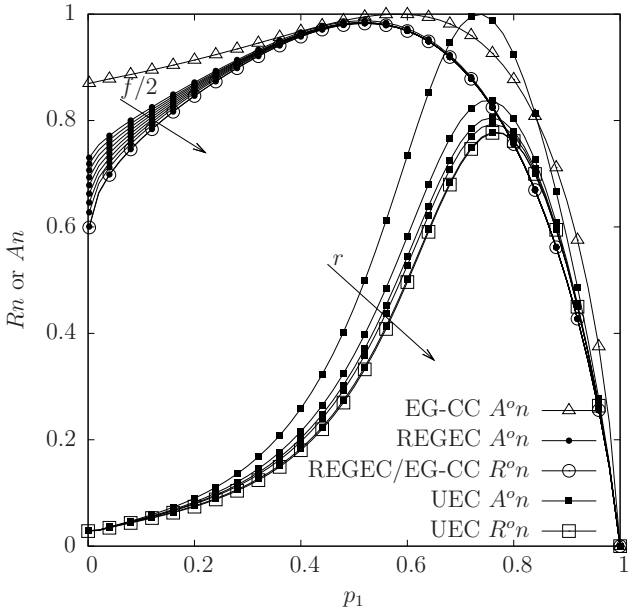


Fig. 10. Plots of $R^{\text{o}}n$ and $A^{\text{o}}n$ that are obtained for the REGEC scheme, EG-CC scheme and UEC scheme, in the case where the symbol values of $\mathbf{d}$ obey a finite Zeta-like distribution having the parameter p_1 and cardinality $L = 1000$. Here, R^{o} is the coding rate, A^{o} is the area beneath the inverted EXIT curve and n is the codeword length of the corresponding scheme. The value of $A^{\text{o}}n$ is provided for an REGEC code having $f/2 \in \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$, while the value of $A^{\text{o}}n$ is provided for a UEC code having $r \in \{2, 4, 6, 8, 30\}$.

the second of the above-mentioned conditions is satisfied when the RVs in the vector $\mathbf{D}$ obey the finite Zeta-like distribution of (1) having the cardinality $L = 1000$ and various values for the parameter p_1. This figure plots the REGEC coding rate R^{o} of (15) when multiplied with the REGEC trellis codeword lengths n. Furthermore, Figure 10 plots the product of n and the area A^{o} of (17) beneath the inverted REGEC EXIT curve for the case where the trellis decoder employs $f/2 \in \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$, giving $r \in \{6, 10, 14, 18, 22, 26, 30, 34, 38\}$ states. Note that according to (15) and (17), the area A^o and coding rate R^o are dependent on the symbol entropy H_D, average REG codeword length l_{REG} and trellis codeword length n, but are independent of the codebook design $\mathbb{C}$. Furthermore, the product of the REGEC EXIT chart area A^o and the codeword length n is related to the number of unary states f, as shown in Figure 10. In the case of the H.265 symbol value distribution of Figure 1, we obtain $R^o n = 0.8787$.

Figure 11 plots the discrepancy between $A^{\text{o}}n$ and $R^{\text{o}}n$ for

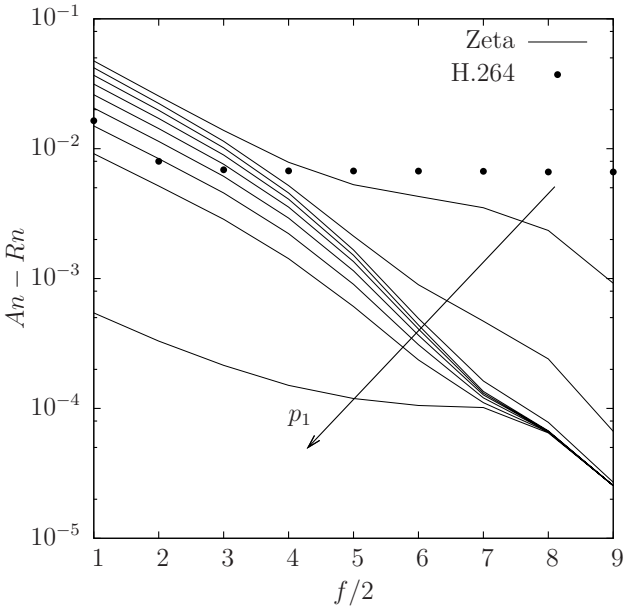


Fig. 11. The discrepancy between $A^{\text{o}}n$ and $R^{\text{o}}n$ that results when REGEC codes having various values of $f/2$ are employed to encode symbol values having finite Zeta-like distributions with the parameters $L = 1000$ and $p_1 \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$, as well as for symbol values obeying the H.265 distribution of Figure 1.

the REGEC code as a function of $f/2$, where the source symbols of $\mathbf{d}$ obey the finite Zeta-like distribution of (1) for a cardinality of $L = 1000$ and for various values for the parameter p_1. Note that in all the scenarios considered, the discrepancy is less than 10^{-1} and becomes less than 10^{-2} when $f/2 \geq 4$, including the case of the H.265 symbol value distribution of Figure 1.

However, the trellis complexity and hence the complexity of REGEC decoding is proportional to the number of states. Our experiments revealed that $f = 2$ and $r = 6$ represents an attractive trade-off between maintaining a low trellis complexity and facilitating near-capacity operation.

C. REGEC codebook candidate selection

In this section, we will discuss the codebook design for an $n = 2$ $r = 6$ REGEC trellis. An $n = 2$ $r = 6$ codebook comprises $r/2 = 3$ codewords, each comprising $n = 2$ bits. Therefore, there are 2^6 possible $n = 2$ $r = 6$ codebooks. However it can be shown that all of these are equivalent to one of the 10 codebooks shown in Table II, which contains no pairs of equivalent codebooks. More specifically,

TABLE II
CANDIDATE REGEC CODEBOOKS $\{\mathbb{C}_i\}_{i=1}^{10}$ FOR $n = 2$ BITS AND $r = 6$ STATES AND THEIR CORRESPONDING FD d_f. FOR FINITE ZETA-LIKE PROBABILITY DISTRIBUTIONS HAVING $L = 1000$ AND VARIOUS VALUES OF p_1, THE NUMBER OF STATES IN THE URC HAVING THE BEST MATCHING EXIT CURVE IS PROVIDED, TOGETHER WITH THE CORRESPONDING E_b/N_o TUNNEL BOUND IN BRACKETS.

candidate codebook		$\mathbb{C}_1$	$\mathbb{C}_2$	$\mathbb{C}_3$	$\mathbb{C}_4$	$\mathbb{C}_5$	$\mathbb{C}_6$	$\mathbb{C}_7$	$\mathbb{C}_8$	$\mathbb{C}_9$	$\mathbb{C}_{10}$
$\mathbf{c}_1$		00	00	00	00	00	00	00	00	00	00
$\mathbf{c}_2$		00	00	00	01	01	01	01	11	11	11
$\mathbf{c}_3$		00	01	11	00	01	10	11	00	01	11
d_f		2	3	2	3	4	4	3	4	3	4
Number of states in URC having best matching EXIT curve and resultant E_b/N_o tunnel bound in dB	p_1 =0.7942	8 (1.6)	4 (2.2)	4 (1.9)	4 (2.1)	2 (2.2)	2 (2.2)	2 (2.2)	4 (1.9)	2 (2.2)	4 (1.8)
	p_1 =0.6	8 (2.2)	2 (2.7)	8 (2.3)	2 (2.7)	2 (2.7)	8 (2.7)	2 (2.8)	8 (2.3)	2 (2.8)	8 (2.3)
	p_1 =0.4	8 (2.2)	2 (2.7)	8 (2.3)	2 (2.7)	2 (2.9)	2 (2.9)	2 (2.7)	8 (2.4)	2 (2.7)	8 (2.3)
	p_1 =0.2	8 (1.8)	2 (2.4)	8 (2)	2 (2.4)	2 (2.7)	2 (2.7)	2 (2.4)	4 (2.4)	2 (2.4)	4 (2.2)

two codebooks are equivalent, if each pairing of codewords within one of the codebooks has the same Hamming Distance (HD) as the corresponding pairing of codewords within the other codebook. Owing to this, two codebooks are equivalent, if one can be transformed into the other by toggling all bits and/or changing the order of the bits in each codeword using the same reordering pattern.

The 10 candidate REGEC codebooks having $n = 2$ $r = 6$ are shown in Table II, where the bits of the codewords have been toggled and reordered in order to minimize the decimal values that are represented by successive codewords. The error correction capability of a codebook may be characterized by the Free Distance (FD) that results at the output of the REGEC trellis encoder[55]. Table II quantifies the FD of each candidate codebook, which was obtained using a brute-force search. As described in Section IV, the REGEC trellis path always starts at the state $m_0 = 1$ and will always end at either state $m_b = 1$ or state $m_b = 2$. Therefore, our brute-force search only needs to consider the free distance between paths that start and end at these states. Owing to this, our experiments revealed that a trellis comprising five stages like that of Figure 8(a) is sufficient for finding the free distance, resulting in only a moderate searching complexity.

Table II suggest that the candidate codebooks $\mathbb{C}_5$, $\mathbb{C}_6$, $\mathbb{C}_8$ and $\mathbb{C}_{10}$ will produce the best error correction capability, since they have the highest FD of 4. However, in iterative decoding schemes it is necessary to separately consider the error correction capability in the turbo cliff and error-floor regions of the Symbol Error Ratio (SER) plot, before the best candidate parametrization can be identified with certainty, as we shall discuss in the following sections.

Note that the FD of on REGEC code remains unaltered if its codebook is extended using the process of Section VI-A, since extension does not change the REGEC-encoding bit vector $\mathbf{z}$ produced for a given REG-encoded bit vector $\mathbf{y}$. Furthermore, depending on the length n of each codeword, the FD of an REGEC code cannot be increased by increasing the number of states r above a particular limit. For example, the largest possible FDs of $n = 2$-bit REGEC codes is 4, regardless of whether $r = 6$ or $r > 6$ number of states are employed. This is because the legitimate transition path set of an r state trellis is a subset of the legitimate transition path set of a trellis having a higher number of possible states $r' > r$. Therefore, we will focus our attention on codebooks corresponding to trellises having $r = 6$ states throughout the remainder of this paper.

D. EXIT charts of the REGEC candidate codebooks and the best matching URCs

As discussed in Section VI-B, the area A^o beneath the inverted REGEC EXIT function and the REGEC coding rate R^o are independent of the codebook design $\mathbb{C}$. However, the shape of the REGEC EXIT curve and therefore its match with the URC EXIT curve does depend on the specific codebook design $\mathbb{C}$. Since the candidate codebooks of Table II are unique with no pair of codebooks that are equivalent to each other, their inverted EXIT curves are all different from each other. Owing to this, different candidate codebooks have inverted EXIT curves that match best with the EXIT curve of URC codes having different parametrizations. In order to investigate this, we plotted the inverted EXIT curves of each candidate REGEC codebook, when used to encode source symbols obeying finite Zeta-like distributions having the cardinality $L = 1000$ and various values for the parameter $p_1 \in \{0.7942, 0.6, 0.4, 0.2\}$. In each case, the resultant EXIT curve was plotted together with the EXIT curves of URC codes having 2, 4 and 8 states. Here generator polynomials of the form $[1, 0, \ldots, 0]$ and feedback polynomials of the form $[1, 1 \ldots, 1]$ were employed, since they are capable of creating open EXIT chart tunnels [13]. The channel E_b/N_0 value was adjusted in each case, until marginally open EXIT chart tunnels were obtained. For each of the cases considered, Table II quantifies the number of states employed by the URC code that creates a marginally open EXIT chart tunnel at the lowest E_b/N_0 value, as well as providing this E_b/N_0 tunnel bound.

Figures 9, 12(a) and 12(b) show the resultant EXIT charts for the cases of using the candidate codebooks $\mathbb{C}_1, \mathbb{C}_8$ and $\mathbb{C}_9$ to encode symbols obeying the finite Zeta-like distribution for $L = 1000$ and $p_1 = 0.7942$. Figures 9, 12(a) and 12(b) also show the corresponding EXIT charts that result in the case, where the symbol probability distribution is unknown in the receiver, as described in Section V-B. The results of Table II show that $\mathbb{C}_1$ is the codebook that facilitates an open EXIT

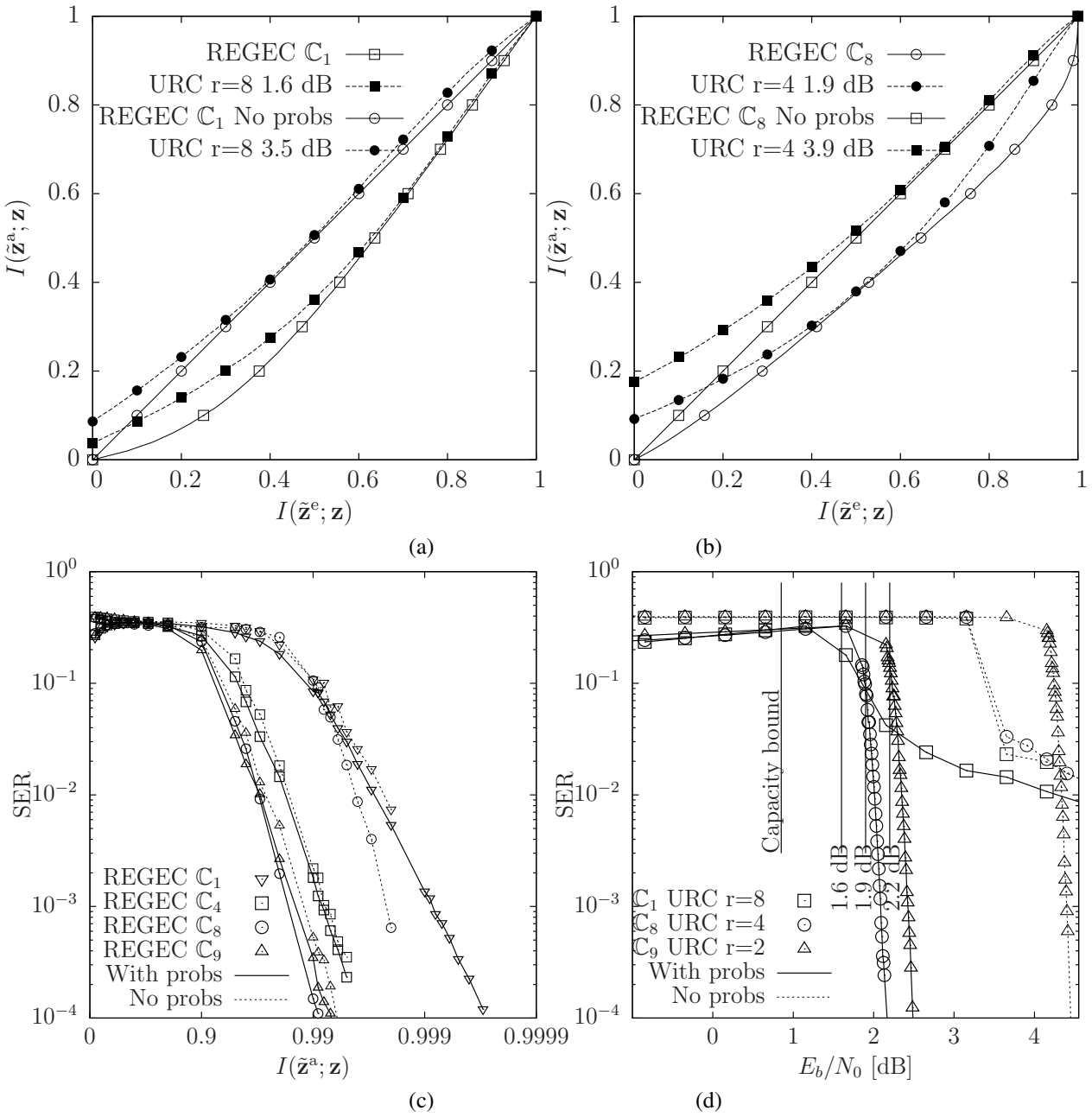


Fig. 12. (a) and (b) EXIT charts of the proposed REGEC scheme. The EXIT curves are provided for REGEC codes employing the $n = 2$-bit $r = 6$-state codebooks $\mathbb{C} \in \{\mathbb{C}_1, \mathbb{C}_8\}$, as well as for a URC having $r \in \{4, 8\}$ states. (c) SER vs E_b/N_0 plot for the REGEC codes employing the $n = 2$-bit $r = 6$-state codebooks $\mathbb{C} \in \{\mathbb{C}_1, \mathbb{C}_8, \mathbb{C}_9\}$, when combined with URC codes having $r \in \{2, 4, 8\}$ states. (d) SER vs $I(\tilde{z}^a; z)$ plot for the REGEC codes employing the $n = 2$-bit $r = 6$-state codebooks $\mathbb{C} \in \{\mathbb{C}_1, \mathbb{C}_4, \mathbb{C}_8, \mathbb{C}_9\}$ when *a priori* LLR vectors $\tilde{\mathbf{z}}^a$ having different MI $I(\tilde{\mathbf{z}}^a; \mathbf{z})$ are provided to the REGEC trellis decoder. In all plots, the symbols of $\mathbf{d}$ obey the finite Zeta-like distribution having $p_1 = 0.7942$ and $L = 1000$. The plots labeled 'No Probs' indicate the case where the source distribution $P(d)$ is unknown to the receiver.

chart tunnel at the lowest E_b/N_0 value. This suggests that $\mathbb{C}_1$ should offer the best performance in the turbo cliff region of the SER plot, since an open EXIT chart tunnel implies that iterative decoding convergence to an ML SER performance can be achieved [13]. However, $\mathbb{C}_1$ may not offer the best performance in the error floor region of the SER plot, as we will investigate in the next section.

E. Error floor analysis

The error correction capability of the candidate REGEC codebooks in the error floor region may be evaluated by considering the SER plots of Figure 12(c). Note that when knowledge of the source probability distribution $P(d)$ is available at the receiver, the candidate codebooks $\mathbb{C}_8$ and $\mathbb{C}_9$ offer steep turbo cliffs at E_b/N_0 values near the corresponding E_b/N_0 tunnel bounds, as predicted by the EXIT charts analysis of Section VI-D. However, the candidate codebook $\mathbb{C}_1$ can be seen to suffer from an error floor, which prevents us from

achieving a low SER at E_b/N_0 values near the corresponding E_b/N_0 tunnel bound of 1.6 dB. This may explained by the observation that the candidate codebook $\mathbb{C}_1$ requires the a $priori$ LLR vector $\tilde{\mathbf{z}}^a$ of Figure 5(a) to have a higher Mutual Information (MI) $I(\tilde{\mathbf{z}}^a;\mathbf{z})$ than $\mathbb{C}_8$ and $\mathbb{C}_9$ require, in order to achieve a low SER, as shown in Figure 12(d). Owing to this, the candidate codebook $\mathbb{C}_1$ requires the iterative decoding process to converge closer towards the (1,1) point of the EXIT chart, which becomes difficult when the interleaver π_1 of Figure 5(a) has only a moderate length [56]. As shown in Figure 12(d), the candidate codebooks $\mathbb{C}_8$ and $\mathbb{C}_9$ require the lowest MIs $I(\tilde{\mathbf{z}}^a;\mathbf{z})$ in order to achieve low SERs.

Meanwhile, the codebooks $\mathbb{C}_5$, $\mathbb{C}_6$, and $\mathbb{C}_7$ have similar SER vs MI curves as $\mathbb{C}_4$ while $\mathbb{C}_2$, $\mathbb{C}_3$ and $\mathbb{C}_{10}$ have similar performance with $\mathbb{C}_1$. Note that the FD-3 codebook $\mathbb{C}_9$ offers better SER performance than several of the other codebooks having FDs of 4. We may speculate that this is because the the error correction capability of a candidate codebook is not only decided by the overall FD but also by the Hamming distances between the codewords that are associated with the transitions in the REGEC trellis having the highest transition probabilities of (11). In the case where the receiver has no knowledge of the source probability distribution $P(d)$, the SER curve of each candidate codebook is degraded, as shown in Figure 12(c). However, this degradation is particularly apparent in the case of $\mathbb{C}_8$, since this causes it to develop an error floor. By contrast, the candidate codebooks $\mathbb{C}_4$, $\mathbb{C}_5$, $\mathbb{C}_6$, $\mathbb{C}_7$ and $\mathbb{C}_9$ do not suffer from an error floor, regardless of whether knowledge of the source probability distribution is available in the receiver while $\mathbb{C}_1$, $\mathbb{C}_2$, $\mathbb{C}_3$ and $\mathbb{C}_{10}$ suffer from error floors for both cases. Overall, we recommend the candidate codebook $\mathbb{C}_9$, since it offers the best performance among the candidate codebooks that never suffer from an error floor. Also, the candidate codebook $\mathbb{C}_9$ works best with the URC inner code having the lowest complexity, namely that employing only $r = 2$ states. Therefore, we employ the candidate codebook $\mathbb{C}_9$ throughout the next section, when we compare the performance of the proposed REGEC scheme with suitably designed benchmarkers.

VII. PERFORMANCE COMPARISON WITH THE BENCHMARKERS

In this section, we compare the proposed REGEC scheme to the EGEC benchmarker of Figures 5(b) [43] and to the UEC and EG-CC benchmarkers of [3]. Like the proposed REGEC schemes, both the EGEC and the UEC benchmarkers constitute examples of JSCCs, while the EG-CC benchmarker represents SSCC. More specifically, the EG-CC benchmarker employs an EG code for source coding, while an iteratively-decoded serial-concatenation of a CC and a URC is employed for separate channel coding. Note that apart from our own previous work, no other JSCC schemes have been design for large-cardinality sources. For example, the Variable Length Error Correction (VLEC) code [57] suffer from excessive complexity for large-cardinality sources, hence preventing a comparison with the proposed REGEC scheme. Again, we used QPSK modulation for transmission over an uncorrelated narrowband Rayleigh fading channel for all schemes, since this is representative of transmissions over realistic wireless channels and because this facilitates direct comparison with the results of [3], [58]. In Section VII-A, we will discuss the parametrization of the REGEC scheme as well as of the three benchmarkers, in order to facilitate fair comparisons. Then we will analyze the SER performance of the proposed REGEC scheme and the three benchmarkers in Section VII-B.

A. Parametrization

Table III provides several parametrizations of the REGEC scheme, which are designed for transmitting symbols that obey the finite Zeta-like distribution of (1). Table III also provides corresponding parametrizations for the three benchmarkers, which offer the same throughput η as our REGEC scheme parametrizations. We parametrize the finite Zeta-like distribution using a cardinality of $L = 1000$ and the parameter of $p_1 \in \{0.7942, 0.6, 0.4, 0.2\}$, which represents a wide selection of the p_1 values shown in Figure 10. Note that the specific value of $p_1 = 0.7942$ is chosen, since it results in the same coding rate for the unary code and the EG code, and hence the same outer coding rate R^o for all schemes considered in this section. Note that, when we have $L \to \infty$, the UEC code becomes impractical for $p_1 = 0.2, 0.4$ and 0.6, since the average unary codeword length becomes infinite in these cases [3]. For finite case of $L = 1000$, the average unary codeword length is more than twice that of the EG code when $p_1 = 0.2$ and 0.4, hence severely degrading the performance of the UEC benchmarker. For this reason, the UEC benchmarker is not considered for these values of p_1. Table III also considers the case of source symbols obeying the H.265 distribution of Figure 1. Note that as described in Section I, the EGEC benchmarker has two parts that must be jointly optimized for each particular source symbol distribution using UEP. More specifically, the puncturing rates R^i for the UEC part and the FLC-CC part must be carefully selected so that they have the same E_b/N_0 tunnel bound [43], as shown in Table III.

For all the schemes considered, we selected codewords comprising $n = 2$ bits when possible, while $n = 3$-bit codewords were selected for the FLC-CC part of the EGEC benchmarker, whenever necessary to achieve the desired effective throughput η for designing the UEP. We selected $r = 6$ states for the proposed REGEC scheme, since this is sufficiently high for imposing only an insignificant amount of capacity loss, as discussed in Section VI-A. Furthermore, we employ the REGEC codebook $\mathbb{C}_9 = [00; 11; 01]$ in order to avoid the error floors that are characterized in Section V-B. Furthermore, we adopt the $r = 4$-state UEC trellis of [3] for both the UEC benchmarker and for the UEC part of the EGEC benchmarker. Meanwhile, we employ an $r = 4$-state CC trellis in both the FLC-CC part of the EGEC benchmarker and in the EG-CC benchmarker, as recommended in [58], [43] and because using higher numbers of states was found to be detrimental in [3]. All of the schemes considered in this section employ URC inner codes, for the sake of facilitating iterative decoding. As discussed in Section VI-D, the selected REGEC codebook $\mathbb{C}_9$ has an EXIT curve that matches best with that of a URC code

having 2 states, shown in Table II. The EGEC, UEC and EG-CC benchmarkers also have EXIT curves that match best with a 2-state URC, since these were found to yield open EXIT chart tunnels at the lowest $E_{\rm b}/N_0$ values in [43]. Therefore, we employ 2-state URCs for the inner codes of all schemes considered in this section. Note that the EGEC, UEC and EG-CC benchmarkers offer fair and natural comparisons with the proposed REGEC scheme, since they all employ simple unary, FLC or EG codewords, as well as trellis-based iterative decoding.

Table III provides the $E_{\rm b}/N_0$ values where the DCMC capacity C becomes equal to the throughput η of each scheme considered. These $E_{\rm b}/N_0$ values represent *capacity bounds*, above which it is theoretically possible to achieve reliable communication, provided that the scheme facilitates near-capacity operation. Furthermore, the specific $E_{\rm b}/N_0$ values, where we have $A^{\rm i} = A^{\rm o}$ are provided for each scheme considered in Table III. These *area bounds* represent the lowest $E_{\rm b}/N_0$ values, where it is theoretically possible to create an open EXIT chart tunnel, provided that the outer and inner EXIT curves have shapes that closely match each other. Note that the discrepancy between the capacity bound and the area bound of each scheme represents an $E_{\rm b}/N_0$ *capacity loss*, as exemplified by Figure 10 for the REGEC, UEC and EG-CC schemes. As in the proposed REGEC code, the EXIT chart area A^o below the inverted UEC curve approaches the UEC coding rate R^o, when the number of states r is increased. By contrast, the EXIT chart area A^o below the inverted EG-CC EXIT curve is not affected by the number of states in the CC trellis, hence resulting in large discrepancies between A^o and R^o, therefore imposing significant amounts of *capacity loss*.

As shown in Table III, the $E_{\rm b}/N_0$ the *capacity loss* of all JSCC schemes is more significant for smaller p_1 values, indicating that trellises having higher numbers of states are required to mitigate *capacity loss* in these cases. However, these capacity losses are smaller than those of the SSCC EG-CC benchmarker, as shown in Table III. For each of the source symbol distributions considered the capacity loss of the REGEC scheme is less than 0.3 dB, which is the smaller than the capacity loss of all the benchmarkers in each case, demonstrating that the proposed REGEC scheme facilitates near-capacity operation. Finally, Table III provides the *tunnel bound* of each scheme, which quantifies the lowest $E_{\rm b}/N_0$ value, where an open EXIT chart tunnel can be created upon employing a two-state accumulator for the URC code, as it was discussed in Section VI-D.

The proposed REGEC schemes facilitate reliable communication at $E_{\rm b}/N_0$ values that exceed the corresponding tunnel bound, provided that the symbol vector $\mathbf{d}$ comprises a sufficiently high number a of symbols. Note that higher $E_{\rm b}/N_0$ values will be required to achieve low SERs, when employing short frames [59]. For all considered values of p_1 as well as for the H.265 distribution, our proposed REGEC scheme offers an open tunnel at the lowest $E_{\rm b}/N_0$ values, facilitating low SERs at low $E_{\rm b}/N_0$ values. At high $E_{\rm b}/N_0$ values, the REGEC scheme will offer the widest open EXIT chart tunnel, requiring fewer decoding iterations to achieve a low SER than the benchmarkers.

Table III also characterizes the complexity of all the schemes considered in this section. Here, the complexity is quantified by the average number of Add, Compare and Select (ACS) operations performed per decoding iteration and per symbol in the vector $\mathbf{d}$. This is justified, since the REGEC trellis decoder UEC trellis decoder, FLC decoder, CC decoder and the URC decoder operate entirely on the basis of addition, subtraction and $\max^*$ operations, which can be further decomposed into ACS operations. All other components in Figure 5 may be considered to have a relatively insignificant complexity [58], [60]. As in [58], we assume that the addition and subtraction operations each require a single ACS operation, while each $\max^*$ operation may be approximated by a look up table operation, which can be completed using five ACS operations [61]. As shown in Table III, the complexity tends to increase as the Zeta distribution parameter p_1 is reduced, which may be explained by the resultant increases in the average codeword lengths l_{REG}, l_{EG} and l_{Unary}. Note that the complexity of the proposed REGEC scheme is higher than those of the benchmarkers, because the REGEC scheme employs an $r = 6$-state trellis, while all benchmarkers employ $r = 4$-state trellises. In order to make fair comparisons in Section VII-B, we will limit the number of decoding iterations performed by the proposed REGEC scheme, so that all schemes operate within the same overall complexity limits. These complexity limits will be chosen to be sufficient for the benchmarker having the lowest complexity to achieve an SER performance that is within 0.1 dB of the performance it can achieve with unlimited complexity. This facilitates a fair comparison by ensuring that the selected complexity limit is not sufficiently high to favor the schemes having the highest complexity, such as the proposed REGEC scheme.

B. SER comparison with the benchmarkers

Figures 13 and 14 characterize the SER performance of the schemes parametrized in Table III. We consider the transmission of source symbol vectors $\mathbf{d}$ comprising $a = 2 \cdot 10^4$ symbols, which we found to be typical of the number of symbols in a H.265[2] slice. Therefore, the SER performance of Figures 13 and 14 may be considered to be achievable without imposing any additional latency in multimedia applications.

As shown in Figure 13, the proposed REGEC scheme facilitates reliable communication within as little as 1.2 dB of the capacity bound and consistently offers the best SER performance for each of the finite Zeta-like distribution p_1 values considered. This consistency is a key benefit of the proposed REGEC scheme, because while it offers only a small gain over the best of the three benchmarkers in each case, the performance of these benchmarkers is particularly inconsistent. More explicitly, while the proposed REGEC scheme offers a gain of 0.4 dB over the UEC benchmarker for $p_1 = 0.7942$, this gain becomes 5 dB for $p_1 = 0.6$, owing to the severe puncturing that the UEC scheme requires in this case [58]. Similarly, while the proposed REGEC scheme offers only a marginal gain over the EGEC benchmarker for $p_1 = 0.6$, this gain becomes 0.8 dB for $p_1 = 0.2$, owing to the severe puncturing of the two parts of the EGEC benchmarker

TABLE III
THE PARAMETERS AND CHARACTERISTIC OF EACH SCHEME CONSIDERED, FOR THE CASE OF SOURCE SYMBOLS OBEYING FINITE ZETA-LIKE DISTRIBUTIONS HAVING $L = 1000$ AND DIFFERENT p_1 VALUES, AS WELL AS FOR THE H.265 DISTRIBUTION OF FIGURE 1.

P_1	Scheme		n	r	R^o	A^o	R^i	η	E_b/N_0 [dB] for $C = \eta$	E_b/N_0 [dB] for $A^i = A^o$	E_b/N_0 [dB] for open tunnel	Complexity
0.7942	REGEC		2	6	0.3834	0.3903	1	0.7669	0.85	1.0	2.2	412
	EGEC	UEC	2	4	0.3746	0.3815	1.0422			1.0	2.4	344
		FLC-CC	3	4	0.2862	0.2953	1.2623					
	UEC		2	4	0.3834	0.4021	1			1.0	2.5	322
	EG-CC		2	4	0.3834	0.4444	1			2.1	3.2	317
0.6	REGEC		2	6	0.4842	0.4877	1	0.9684	1.69	1.8	2.8	662
	EGEC	UEC	2	4	0.4904	0.4910	1			1.8	2.8	530
		FLC-CC	2	4	0.4696	0.4775	1					
	UEC		2	4	0.2482	0.2910	1.9505			2.9	5.7	1009
	EG-CC		2	4	0.4842	0.4995	1			2.0	2.9	510
0.4	REGEC		2	6	0.4789	0.4845	1	0.9578	1.65	1.8	2.7	1147
	EGEC	UEC	2	4	0.4735	0.4783	1			2.0	2.9	907
		FLC-CC	2	4	0.4876	0.4930	1					
	EG-CC		2	4	0.4789	0.4845	1			1.8	2.7	884
0.2	REGEC		2	6	0.4231	0.4401	1	0.8462	1.18	1.4	2.4	2048
	EGEC	UEC	2	4	0.3678	0.3956	1.1512			1.7	3.0	1596
		FLC-CC	3	4	0.3301	0.3390	1.2814					
	EG-CC		2	4	0.4231	0.4584	1			1.8	2.9	1578
H.265	REGEC		2	6	0.4393	0.4486	1	0.8786	1.3	1.5	2.6	724
	EGEC	UEC	2	4	0.4639	0.4652	1			1.8	2.9	588
		FLC-CC	2	4	0.3862	0.3955	1					
	UEC		2	4	0.3480	0.4249	1.2624			3.1	4.7	715
	EG-CC		2	4	0.4393	0.4961	1			2.4	3.3	558

in order to achieve UEP [58], as described in Section I. Note that the EGEC scheme has worse performance than the SSCC EG-CC benchmarker for $p_1 \in \{0.2, 0.4\}$. In the case of $p_1 = 0.2$, this may also be attributed to the severe puncturing invoked for UEP. In the case of $p_1 \in \{0.4, 0.6\}$, UEP does not improve the performance of the EGEC benchmarker, beyond that of the Equal Error Protection (EEP). Since our proposed REGEC scheme does not have two parts that must be carefully balanced, it does not suffer from these problems. Similarly, while the proposed REGEC scheme offers only a marginal gain over the EG-CC benchmarker for $p_1 = 0.4$, this gain becomes 0.6 dB for $p_1 = 0.2$ and 0.9 dB for $p_1 = 0.7942$, as shown in Figure 13.

In the case where the source symbols obey the H.265 distribution of Figure 1, our REGEC scheme offers a gain of 0.7 dB over the SSCC EG-CC benchmarker, as shown in Figure 14 -. Furthermore, our REGEC scheme offers 0.3 dB gain over the EGEC benchmarker, where UEP does not improve the performance of the EGEC benchmarker in this scenario. The UEC benchmarker has the worst performance of all the schemes considered in this scenario, owing to the severe puncturing that it requires to achieve the same effective throughput as the other schemes.

Note that since the SER results of Figures 13 and 14 offer fair comparisons in terms of complexity and effective throughput, the gains offered by our proposed REGEC scheme are obtained for free, with no cost in terms of transmit-duration, transmit-bandwidth, transmit-energy or decoding complexity. Therefore, a gain of say 0.9 dB may be deemed significant, particularly since it is achieved in the extreme vicinity of the DCMC capacity bound, namely within about 1.5 dB. This is achieved by mitigating the capacity loss, which is inherent in SSCC and which limits the performance of other JSCC schemes. Since these gains are associated with the improvements offered by the REGEC code over the benchmarker SSCC and JSCC codes, similar gains may be expected when combining them with any other channel codes, modulation schemes or channels.

Note that throughout our discussions above, it was assumed that the receiver of the proposed REGEC scheme has knowledge of the average REG codeword length l. Furthermore, it was assumed that the decoder has knowledge of the probabilities of occurrence $P(d)$. However, Figure 13 and 14 show that when the channel SNR is sufficiently high, the proposed REGEC receiver facilitates a low SER, even if it does not have any knowledge of the symbol probabilities $P(d)$. The symbol probabilities may be estimated by storing a sufficient number of symbol vectors $\hat{\mathbf{d}}$, in order to heuristically estimate the required information, hence facilitating near-capacity communication for the subsequent symbol vectors.

VIII. CONCLUSIONS

In this paper, we have proposed a novel REGEC code for the near-capacity transmission of symbol values that are randomly selected from a source set having a large or infinite cardinality. Our REGEC code comprises a novel REG source code and a novel trellis code, which facilitates joint source and channel coding. In contrast to the UEC code previously proposed for the same purpose, our REGEC code is a universal code, facilitating the transmission of symbol values that are randomly selected using any mototonic probability distribution. On the other hand, in contrast to the EGEC code previously proposed for the same purpose, our REGEC code has a simple structure, which solves the delay, synchronization and computational complexity problems associated with the two parts of the EGEC code. In particular, the EGEC code must be specially parametrized for operation with the particular source

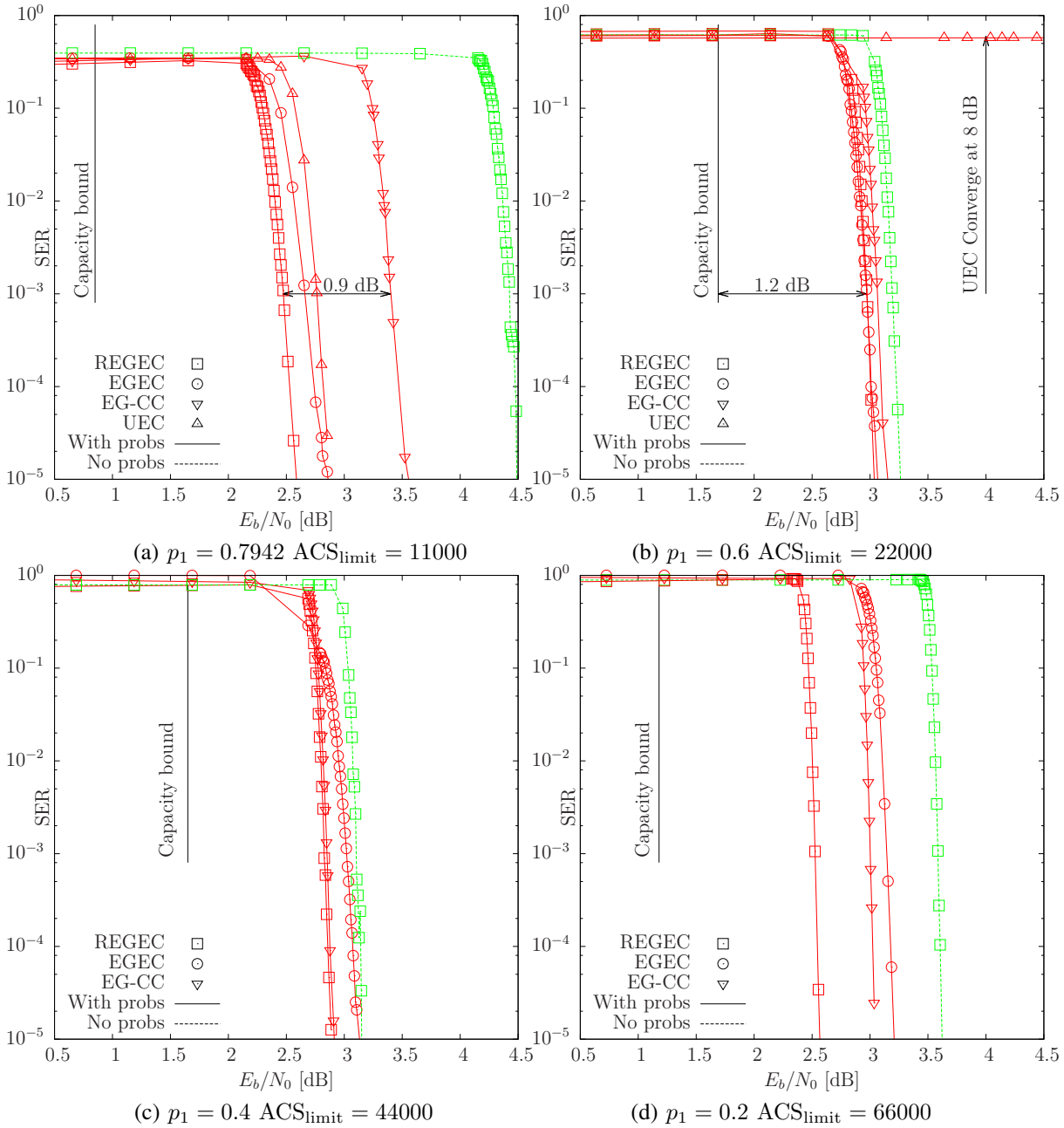


Fig. 13. The SER performance of the REGEC scheme and the UEC, EGEC and EG-CC benchmarkers of Figure 5, when transmitting frames comprising $a = 2 \cdot 10^4$ symbols that obey the finite Zeta-like distribution having the cardinality of $L = 1000$ and various p_1 values, using QPSK modulation is employed for transmission over an uncorrelated narrowband Rayleigh fading channel. Iterative decoding continues until $\text{ACS}_{\text{limit}}$ number of ACS operation have been performed per symbol in the vector $\mathbf{d}$.

distribution, preventing its application for unknown or non-stationary sources. By contrast, the proposed REGEC code can be applied for any distribution without requiring special parametrization.

In some practical scenarios where the source symbols obey particular finite Zeta-like probability distributions, our REGEC scheme is shown to offer gains of up to 0.9 dB over the best benchmarkers, when QPSK modulation is employed for transmission over an uncorrelated narrowband Rayleigh fading channel. In the scenario where the source symbols obey the H.265 distribution, our REGEC scheme is shown to offer a gain of 0.7 dB over the SSCC benchmarker, when QPSK modulation is employed for transmission over an uncorrelated narrowband Rayleigh fading channel. These gains are achieved for free, without increasing the required transmit-duration, transmit-bandwidth, transmit-energy or decoding complexity. We consider these gains to be significant, since they are achieved within the extreme vicinity of the DCMC capacity, namely within 1.4 dB. This is achieved by mitigating the capacity loss inherent in SSCC, which limits the performance of other JSCC schemes. Since these gains are associated with the improvements offered by the REGEC code over the benchmarker SSCC and JSCC codes, similar gains may be expected when combining with any other channel codes,

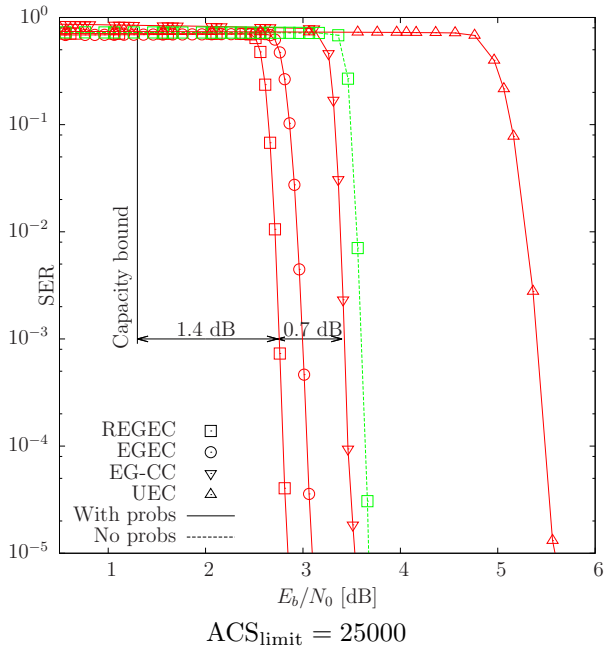


Fig. 14. The SER performance of the REGEC scheme and the UEC, EGEC and EG-CC benchmarkers of Figure 5, when transmitting frames comprising $a = 2 \cdot 10^4$ symbols that obey the H.265 distribution, using QPSK modulation is employed for transmission over an uncorrelated narrowband Rayleigh fading channel. Iterative decoding continues until $\mathrm{ACS}_{\mathrm{limit}}$ number of ACS operation have been performed per symbol in the vector $\mathbf{d}$.

modulation schemes or channels.

Our future work will consider the integration of the proposed REGEC code into a practical video codec, such as H.264 or H.265. This may be achieved by replacing all EG codewords with REG codewords. Since the iteration decoding process performs best when the interleaver length is long, it may be necessary to modify the video codec in order to keep all REG-encoded bits together within each frame. The resultant bit vector $\mathbf{y}$ can then be trellis encoded, interleaved and URC encoded, as shown in Figure 5(a). Following modulation, transmission and demodulation, iterative decoding may be employed to recover the REG-encoded LLRs $\tilde{\mathbf{y}}^{\mathrm{p}}$. These LLRs may then be converted into symbols, or the soft information of these LLRs may be exploited to aid error concealment.

APPENDIX

DERIVATION OF (11)

The method of [3, Appendix] may be used to derive the transition probabilities $P(m, m')$ of (11) by observing the expected number of transitions of each type when encoding symbols in the vector $\mathbf{d}$. More specifically, a transition from a unary state $m' \in \{1, 2, \ldots, f\}$ to a FLC state $m = m' + f$ will occur for each symbol in the vector $\mathbf{d}$ satisfying $d_i \geq 2^{\lceil m'/2 \rceil}$. The number of symbols in the vector $\mathbf{d}$ that satisfy this conditions has an expected value of

$$\begin{aligned} & \tfrac{a}{2} \sum_{d=2^{\lceil \frac{m'}{2} \rceil}}^{L} P(d) \\ = \; & \tfrac{a}{2} \left[\sum_{d=1}^{L} P(d) - \sum_{d=1}^{2^{\lceil \frac{m'}{2} \rceil}-1} P(d) \right] \\ = \; & \tfrac{a}{2} \left[1 - \sum_{d=1}^{2^{\lceil \frac{m'}{2} \rceil}-1} P(d) \right]. \end{aligned}$$

Therefore we may expect half of this number of the transitions in the path $\mathbf{m}$ to be of each of the above-mentioned types on average, since the trellis is symmetric and the transitions where $\mathrm{odd}(m') = 0$ and $\mathrm{odd}(m') = 1$ are equiprobable.

Similarly, a transition from a unary state $m' \in \{1, 2, \ldots, f\}$ to a unary state $m = 1 + \mathrm{odd}(m')$ will occur for each symbol in the vector $\mathbf{d}$ satisfying $\lfloor \log_2(d_i) \rfloor = \lceil m'/2 \rceil$. We can expect $\left[\frac{a}{2} \cdot \sum_{d=2^{\lceil m'/2 \rceil - 1}}^{2^{\lceil m'/2 \rceil}-1} P(d) \right]$ of the symbols in the vector $\mathbf{d}$ to satisfy these conditions and therefore we can expect half of this many of the transitions in the path $\mathbf{m}$ to be of each of the above-mentioned types.

Furthermore, a transition from an FLC state $m_{j-1} \in \{f+1, f+2, \ldots, 2f-2\}$ to a unary state $m = \ddot{d} - y_j + 2y_j \cdot \mathrm{odd}(m')$ will occur for each symbol in the vector $\mathbf{d}$ satisfying $d_i \geq 2^{\lceil (m'-f)/2 \rceil}$. We can expect $\left[\frac{a}{2} \cdot \sum_{x=\ddot{x}+1}^{\lfloor \log_2(L) \rfloor 2^{\ddot{x}}-1} \sum_{\ddot{t}=0}^{} \mathrm{odd}(\ddot{t}+1+y_i) \sum_{d=2^x+(\ddot{t}+1)\frac{2^x}{2^{\ddot{x}}}-1}^{2^x+(\ddot{t}+1)\frac{2^x}{2^{\ddot{x}}}-1} P(d) \right]$ of the symbols in the vector $\mathbf{d}$ to satisfy these conditions and therefore we can expect half of this many of the transitions in the path $\mathbf{m}$ to be of each of the above-mentioned types.

Moreover, a transition from a holding state $m' \in \{2f-1, 2f\}$ to a holding state $m = m' + y_j \cdot (2 \cdot \mathrm{odd}(m') + 1) + 2 \cdot \mathrm{odd}(y_j + 1)$ will occur for each symbol in the vector $\mathbf{d}$ satisfying $d_i \geq 2^{\lceil f/2 \rceil}$. We can expect $\left[\frac{a}{2} \cdot \sum_{x=f}^{\lfloor \log_2(L) \rfloor} \sum_{\mathring{x}=f}^{x} \sum_{\mathring{t}=0}^{2^{\mathring{x}}-1} \mathrm{odd}(\mathring{t}+1+y_i) \sum_{d=2^x+\mathring{t}\frac{2^x}{2^{\mathring{x}}}}^{2^x+(\mathring{t}+1)\frac{2^x}{2^{\mathring{x}}}-1} P(d) \right]$ of the symbols in the vector $\mathbf{d}$ to satisfy these conditions and therefore we can expect half of this many of the transitions in the path $\mathbf{m}$ to be of each of the above-mentioned types. In addition, a transition from a holding state $m' \in \{2f+1, 2f+2\}$ to a unary state $m = 1 + \mathrm{odd}(m')$ will occur for each symbol in the vector $\mathbf{d}$ satisfying $d_i \geq 2^{\lceil f/2 \rceil}$. We can expect $\left[\frac{a}{2} \cdot \left[1 - \sum_{d=1}^{2^{f/2}-1} p(d) \right] \right]$ of the symbols in the vector $\mathbf{d}$ to satisfy these conditions and therefore we can expect half of this many of the transitions in the path $\mathbf{m}$ to be of each of the above-mentioned types.

Finally, each symbol in the vector $\mathbf{d}$ satisfying $d_i \geq 2^{\lceil f/2 \rceil}$ will yield $\log_2(d_i) - f/2$ transitions from a holding state $m' \in \{2f+1, 2f+2\}$ to a holding state $m = m' - 2$. Therefore, the number of transitions in the path $\mathbf{m}$ that can be expected

to be of each of the above-motioned types is given by

$$\begin{aligned} & \frac{a}{2}\sum_{d=2^{f/2}}^{L} P(d)(\lfloor \log_2(d) \rfloor - \tfrac{f}{2}) \\ = \; & \frac{a}{2}\left[\sum_{d=1}^{L} P(d)(\lfloor \log_2(d) \rfloor - \tfrac{f}{2}) - \sum_{d=1}^{2^{f/2}-1} P(d)(\lfloor \log_2(d) \rfloor - \tfrac{f}{2})\right] \\ = \; & \frac{a}{2}\left[l_1 - \tfrac{f}{2} - \sum_{d=1}^{2^{f/2}-1} P(d)(\lfloor \log_2(d) \rfloor - \tfrac{f}{2})\right], \end{aligned}$$

where l_1 is the average length of the unary codeword Unary(x_i), as described in Section III.

Dividing the result for all cases by the expected number of transition in the path $\mathbf{m}$, namely al, yields the transition probability given in (11).

REFERENCES

[1] "Advanced video coding for generic audiovisual services," tech. rep., ITU-T Std. H.264, March 2005.

[2] ITU-T, *Recommendation H.265: High efficiency video coding*, June 2015.

[3] R. Maunder, W. Zhang, T. Wang, and L. Hanzo, "A unary error correction code for the near-capacity joint source and channel coding of symbol values from an infinite set," *IEEE Transactions on Communications*, vol. 61, pp. 1977–1987, May 2013.

[4] N. L. Johnson, A. W. Kemp, and S. Kotz, *Univariate Discrete Distributions*. New York, NY, USA: John Wiley & Sons, 2005.

[5] R. Gallager and D. van Voorhis, "Optimal source codes for geometrically distributed integer alphabets," *IEEE Transactions on Information Theory*, vol. 21, pp. 228–230, March 1975.

[6] P. Elias, "Universal codeword sets and representations of the integers," *IEEE Transactions on Information Theory*, vol. 21, pp. 194–203, March 1975.

[7] M. Wien, *High Efficiency Video Coding: coding Tools and Specification*. Springer, 2015.

[8] C. E. Shannon, *Mathematical Theory of Communication*. University of Illinois Press, 1963.

[9] B. Ryabko and J. Rissanen, "Fast adaptive arithmetic code for large alphabet sources with asymmetrical distributions," in *Proc. IEEE International Symposium on Information Theory*, (Lausanne, Switzerland), p. 319, June 2002.

[10] J. Ziv and A. Lempel, "Compression of indivdual sequences via variable-rate coding," *IEEE Transactions on Information Theory*, vol. 24, pp. 530–536, Sept. 1978.

[11] D. Mackay and R. Neal, "Near Shannon limit performance of low density parity check codes," *Electronic Letter*, vol. 32, pp. 457–458, Aug. 1996.

[12] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes (1)," in *Proceedings of the International Conference on Communications*, vol. 2, (Geneva, Switzerland), pp. 1064–1070, May 1993.

[13] L. Hanzo, R. Maunder, J. Wang, and L.-L. Yang, *Near-Capacity Variable Length Coding*. Chichester, UK: Wiley, 2010.

[14] C. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, pp. 379–423, July 1948.

[15] R. Fano, *The transmission of information*. Massachusetts Institute of Technology, Research Laboratory of Electronics, 1949.

[16] D. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, pp. 1098–1101, Sept 1952.

[17] N. Abramson, *Information Theory and Coding*. New York, USA: McGraw-Hill, 1966.

[18] S. Golomb, "Run-length encodings (corresp.)," *IEEE Transactions on Information Theory*, vol. 12, pp. 399–401, Jul 1966.

[19] J. Teuhola, "A compression method for clustered bit-vectors," *Information processing letters*, vol. 7, no. 6, pp. 308–311, 1978.

[20] A. Apostolico and A. Fraenkel, "Robust transmission of unbounded strings using fibonacci representations," *IEEE Transactions on Information Theory*, vol. 33, pp. 238–245, Mar 1987.

[21] S. Even and M. Rodeh, "Economical Encoding of Commas Between Strings.," *ommunications of the ACM*, vol. 21, no. 4, pp. 315–317, 1978.

[22] Q. Stout, "Improved prefix encodings of the natural numbers (Corresp.).," *IEEE Transactions on Information Theory*, vol. 26, no. 5, pp. 607–609, 1980.

[23] A. S. Fraenkel and S. T. Klein, "Robust Universal Complete Codes for Transmission and Compression.," *Discrete Applied Mathematics*, vol. 64, no. 1, pp. 31–55, 1996.

[24] R. Hamming, "Error detecting and error correcting codes," *Bell System Technical Journal*, vol. 29, pp. 147–160, 1950.

[25] I. Reed, "A class of multiple-error-correcting codes and the decoding scheme," *Transactions of the IRE Professional Group on Information Theory*, vol. 4, pp. 38–49, September 1954.

[26] J. Wozencraft, "Sequential decoding for reliable communication," *IRE National Convention Record*, vol. 5, pt.2, pp. 11–25, 1957.

[27] R. Bose and D. Ray-Chaudhuri, "On a class of error correcting binary group codes," *Information and Control*, vol. 3, pp. 68–79, March 1960.

[28] A. Hocquenghem, "Codes correcteurs d'Erreurs," *Chiffres (Paris)*, vol. 2, pp. 147–156, September 1959.

[29] I. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the society for industrial and applied mathematics*, vol. 8, pp. 300–304, June 1960.

[30] R. Gallager, "Low-Density Parity-Check Codes," *IEEE Transactions on Information Theory*, pp. 21–28, 1962.

[31] L. Bahl, C. Cullum, W. Frazer, and F. Jelinek, "An efficient algorithm for computing free distance (corresp.)," *IEEE Transactions on Information Theory*, vol. 18, pp. 437–439, May 1972.

[32] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimising symbol error rate," *IEEE Transactions on Information Theory*, vol. 20, pp. 284–287, March 1974.

[33] J. Wolf, "Efficient Maximum Likelihood Decoding of Linear Block Codes Using a Trellis," *IEEE Transactions on Information Theory*, pp. 76–80, 1978.

[34] W. Koch and A. Baier, "Optimum and Sub-Optimum Detection of Coded Data Disturbed by Time-Varying Intersymbol Interference," in *IEEE Global Telecommunications Conference, 1990*, pp. 1679–1684, 1990.

[35] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal map decoding algorithms operating in the log domain," in *IEEE International Conference on Communications,*, vol. 2, (Seattle, USA), pp. 1009–1013 vol.2, Jun 1995.

[36] T. Richardson, M. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 47, pp. 619–637, Feb 2001.

[37] M. Luby, "Lt codes,," in *Proceedings of 43rd Annunal IEEE Symposium Foundations of Computer Science,*, (Vancouver, BC, Canada), pp. 1009–1013 vol.2, November 2002.

[38] A. Shokrollahi, "Raptor codes," *IEEE Transactions on Information Theory*, vol. 52, pp. 2551–2567, June 2006.

[39] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Transactions on Information Theory,*, vol. 55, pp. 3051–3073, July 2009.

[40] N. Bonello, R. Zhang, S. Chen, and L. Hanzo, "Reconfigurable rateless codes," *IEEE Transactions on Wireless Communications*, vol. 8, pp. 5592–5600, November 2009.

[41] R. Maunder, "A fully-parallel turbo decoding algorithm," *IEEE Transactions on Communications*, vol. 63, pp. 2762–2775, Aug 2015.

[42] J. Massey, "Joint source and channel coding," *Communication Systems and Random Process Theory*, pp. 279–293, December 1978.

[43] T. Wang, W. Zhang, R. Maunder, and L. Hanzo, "Near-capacity joint source and channel coding of symbol values from an infinite source set using elias gamma error correction codes," *IEEE Transactions on Communications*, vol. 62, pp. 280–292, January 2014.

[44] D. Divsalar, S. Dolinar, and F. Pollara, "Serial concatenated trellis coded modulation with rate-1 inner code," in *Global Telecommunications Conference, 2000. GLOBECOM '00. IEEE*, vol. 2, pp. 777–782 vol.2, 2000.

[45] L. Hanzo, S. Ng, T. Keller, and W. T. Webb, *Quadrature amplitude modulation: From basics to adaptive trellis-coded, turbo-equalised and space-time coded OFDM, CDMA and MC-CDMA systems*. IEEE Press-John Wiley, 2004.

[46] X. Li and J. A. Ritcey, "Bit-interleaved coded modulation with iterative decoding," *IEEE Communications Letters*, vol. 1, pp. 169–171, Nov 1997.

[47] M. Tuchler, "Convergence prediction for iterative decoding of threefold concatenated systems," in *IEEE Global Telecommunications Conference*, vol. 2, (Taipei, Taiwan), pp. 1358–1362, November 2002.

[48] R. Maunder and L. Hanzo, "Iterative decoding convergence and termination of serially concatenated codes," *IEEE Transactions on Vehicular Technology*, vol. 59, pp. 216–224, January 2010.

[49] S. Ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Transactions on Communications*, vol. 49, pp. 1727–1737, October 2001.

[50] J. Kliewer, N. Görtz, and A. Mertins, "Iterative source-channel decoding with Markov random field source models," *IEEE Transactions on Signal Processing*, vol. 54, pp. 3688–3701, October 2006.

[51] D. Divsalar, H. Jin, and R. McEliece, "Coding theorems for 'turbo-like' codes," in *Proc. Allerton Conf. on Communications, Control and Computing*, pp. 201–210, September 1998.

[52] A. Ashikhmin, G. Kramer, and S. Ten Brink, "Extrinsic information transfer functions: model and erasure channel properties," *IEEE Transactions on Information Theory*, vol. 50, pp. 2657–2673, November 2004.

[53] W. Zhang, Y. Jia, X. Meng, M. Brejza, R. Maunder, and L. Hanzo, "Adaptive iterative decoding for expediting the convergence of unary error correction codes," *IEEE Transactions on Vehicular Technology*, vol. 64, pp. 621–635, Feb 2015.

[54] J. Proakis, *Digital Communications*. New York, USA: McGraw-Hill, 3rd ed., 1995.

[55] V. Buttigieg and P. Farrell, "Variable-length error-correcting codes," *IEE Proceedings Communications*, vol. 147, pp. 211–215, Aug 2000.

[56] J. Hokfelt, O. Edfors, and T. Maseng, "A turbo code interleaver design criterion based on the performance of iterative decoding," *IEEE Communications Letters*, vol. 5, pp. 52–54, Feb 2001.

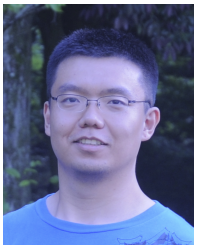
[57] R. Maunder and L. Hanzo, "Genetic algorithm aided design of component codes for irregular variable length coding," *IEEE Transactions on Communication*, vol. 57, pp. 1290–1297, May 2009.

[58] W. Zhang, R. Maunder, and L. Hanzo, "On the complexity of unary error correction codes for the near-capacity transmission of symbol values from an infinite set," in *Proc. IEEE Wireless Communications and Networking Conference*, (Shanghai, China), 2013 http://eprints.soton.ac.uk/344059/.

[59] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Transactions on Information Theory*, vol. 42, pp. 429–445, March 1996.

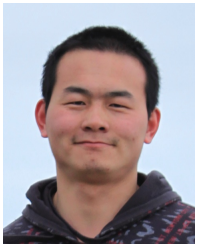
[60] M. Adrat, R. Vary, and J. Spittka, "Iterative Source-Channel Decoder Using Extrinsic Information from Softbit-Source Decoding," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 4, (Salt Lake City, UT, USA), pp. 2653–2656, May 2001.

[61] L. Li, R. Maunder, B. Al-Hashimi, and L. Hanzo, "A low-complexity turbo decoder architecture for energy-efficient wireless sensor networks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, pp. 14–22, Jan 2013.



Tao Wang received the B.S. degree in information engineering from the University of Science and Technology of Beijing (USTB), Beijing, China, in 2006. He received M.Sc. degree in communication from University of Southampton, Southampton, U.K in 2008. He is currently working toward the Ph.D. degree with the Communications Research Group, Electronics and Computer Science, University of Southampton, Southampton, UK. His current research interests include joint source/channel coding and distributed video coding.

Matthew F. Brejza (http://users.ecs.soton.ac.uk/mfb2g09) received a first class honors BEng in Electronic Engineering from the University of Southampton, UK, in July 2012, where he is currently working toward the Ph.D. degree with the Communications Research Group, School of Electronics and Computer Science. His research interests include flexible hardware implementation, channel coding and their applications in low power data communications.



Wenbo Zhang received the M.E. degree in Information and Communication Engineering from the University of Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 2011. He is currently working toward the Ph.D. degree with the Communications Research Group, Electronics and Computer Science, University of Southampton, Southampton, UK. His current research interests include joint source/channel coding and variable length coding.

Robert G. Maunder (http://users.ecs.soton.ac.uk/rm) has studied with Electronics and Computer Science, University of Southampton, UK, since October 2000. He was awarded a first class honors BEng in Electronic Engineering in July 2003, as well as a PhD in Wireless Communications and a lectureship in December 2007. Rob's research interests include joint source/channel coding, iterative decoding, irregular coding and modulation techniques. He has published a number of IEEE papers in these areas.

Lajos Hanzo (http://www-mobile.ecs.soton.ac.uk) FREng, FIEEE, FIET, Fellow of EURASIP, DSc received his degree in electronics in 1976 and his doctorate in 1983. In 2009 he was awarded an honorary doctorate by the Technical University of Budapest and in 2015 by the University of Edinburgh. During his 40-year career in telecommunications he has held various research and academic posts in Hungary, Germany and the UK. Since 1986 he has been with the School of Electronics and Computer Science, University of Southampton, UK, where he holds the chair in telecommunications. He has successfully supervised 110 PhD students, co-authored 20 John Wiley/IEEE Press books on mobile radio communications totalling in excess of 10 000 pages, published 1600+ research contributions at IEEE Xplore, acted both as TPC and General Chair of IEEE conferences, presented keynote lectures and has been awarded a number of distinctions. Currently he is directing a 60-strong academic research team, working on a range of research projects in the field of wireless multimedia communications sponsored by industry, the Engineering and Physical Sciences Research Council (EPSRC) UK, the European Research Council's Advanced Fellow Grant and the Royal Society's Wolfson Research Merit Award. He is an enthusiastic supporter of industrial and academic liaison and he offers a range of industrial courses. He is also a Governor of the IEEE VTS. During 2008 - 2012 he was the Editor-in-Chief of the IEEE Press and a Chaired Professor also at Tsinghua University, Beijing. His research is funded by the European Research Council's Senior Research Fellow Grant. Lajos has 25 000+ citations. For further information on research in progress and associated publications please refer to http://www-mobile.ecs.soton.ac.uk