

Approximation of DAC Codeword Distribution for Equiprobable Binary Sources along Proper Decoding Paths

Yong Fang

Abstract

Distributed Arithmetic Coding (DAC) is an effective implementation of Slepian-Wolf coding, especially for short data blocks. To research its properties, the concept of DAC codeword distribution along proper and wrong decoding paths has been introduced. For DAC codeword distribution of equiprobable binary sources along proper decoding paths, the problem was formatted as solving a system of functional equations. However, up to now, only one closed form was obtained at rate 0.5, while in general cases, to find the closed form of DAC codeword distribution still remains a very difficult task. This paper proposes three kinds of approximation methods for DAC codeword distribution of equiprobable binary sources along proper decoding paths: numeric approximation, polynomial approximation, and Gaussian approximation. Firstly, as a general approach, a numeric method is iterated to find the approximation to DAC codeword distribution. Secondly, at rates lower than 0.5, DAC codeword distribution can be well approximated by a polynomial. Thirdly, at very low rates, a Gaussian function centered at 0.5 is proved to be a good and simple approximation to DAC codeword distribution. A simple way to estimate the variance of Gaussian function is also proposed. Plenty of simulation results are given to verify theoretical analyses.

Index Terms

Distributed Source Coding, Slepian-Wolf Coding, Distributed Arithmetic Coding, Codeword Distribution.

This research was supported by NAFU Talent Fund Z111020901.

Y. Fang is with the College of Information Engineering, Northwest A&F University, Shaanxi Yangling 712100, China (email: yfang79@gmail.com).

I. INTRODUCTION

Consider the problem of Slepian-Wolf Coding (SWC) with decoder Side Information (SI), i.e. the encoder compresses discrete source X in the absence of Y , discretely-correlated SI. Slepian-Wolf theorem states that lossless compression is achievable at rates $R \geq H(X|Y)$ [1], where $H(X|Y)$ is the conditional entropy of X given Y . Conventionally, channel codes, e.g., turbo codes [2] or Low-Density Parity-Check (LDPC) codes [3], are used to implement the SWC.

Ever since a long time ago, Arithmetic Coding (AC) has been proposed as the successor of Huffman coding to implement source coding and shows near-entropy performance [4], [5], [6]. At the same time of high compression efficiency, the AC increases computational complexity and noise sensitivity of the bitstream. To reduce computational complexity, Quasi-Arithmetic Coding (QAC) has been introduced in [7]. To fight against noise sensitivity, redundancies are usually reinjected into the bitstream by different means. In [8], redundancies are reinjected into the bitstream in the form of parity-check bits. In [9], markers are inserted at known positions in the sequence of source symbols. In [10], forbidden intervals are exploited for error detection. These approaches fall into the so-called Error Detecting AC (EDAC). The EDAC can be coupled with Automatic Repeat reQuest (ARQ) [11], [12], [13] or channel codes [13] to support error correction. To realize Error Correcting AC (ECAC), sequential decoding of arithmetic codes with forbidden intervals is proposed in [14], whose complexity is reduced in [15] by using Trellis-Coded Modulation (TCM) and List Viterbi decoding Algorithm (LVA). A soft decoding procedure is described in [16], whose counterpart for QAC appears in [17]. The Maximum A Posteriori (MAP) decoding procedure is proposed and applied to image transmission [18], [19], [20], [21].

Recently, the AC is also applied to implement the SWC. One approach is to allow overlapped intervals, which mirrors the work in [10]. Such examples include Distributed Arithmetic Coding (DAC) [22], [23] and Overlapped Quasi-Arithmetic Coding (OQAC) [24]. Another approach is to puncture some bits of AC bitstream, e.g. Punctured Quasi-Arithmetic Coding (PQAC) [25], which mirrors the work in [9]. There are also some variants of the DAC. The symmetric SWC is implemented by the time-shared DAC (TS-DAC) [26]. The rate-compatible DAC is proposed in [27]. Furthermore, decoder-driven adaptive DAC [28] is proposed to estimate source probabilities on-the-fly.

We note that the ECAC and the DAC in fact generalize the classic AC in reverse directions. The ECAC encodes source X at rates $R \geq H(X)/C \geq H(X)$ by introducing forbidden intervals, where C is channel capacity. The forbidden intervals, corresponding to forbidden symbols, lead to a longer codeword due to narrowed final interval and also inject redundancies into the resulting bitstream. The decoder jointly exploits both the received bitstream and known channel parameters to reconstruct source X . The DAC encodes source X at rates $H(X|Y) \leq R \leq H(X)$ by allowing overlapped intervals. The overlapped intervals, corresponding to ambiguous symbols, lead to a shorter codeword due to enlarged final interval and also induce ambiguities in the resulting bitstream. A soft joint decoder exploits both the received bitstream and Y to reconstruct X .

Though it is well-known that the classic AC can achieve source entropy $H(X)$ theoretically, it is not clear whether the DAC can achieve conditional entropy $H(X|Y)$. If no, what is the performance limit of the DAC? Is it possible to improve its performance? If yes, how to realize it? Intuitively, before answering these questions, one may need to know how many branches will be generated during the DAC decoding. In addition, it may also be helpful to know the distribution of Hamming distances between decoding branches and source X .

As the first step, to analyze the properties of the DAC, [29] introduces the concept of codeword distribution, which seems promising for answering these questions. DAC codeword distribution is a function defined over interval $[0, 1)$. For equiprobable binary sources, both codeword distribution along proper decoding paths and codeword distribution along wrong decoding paths are researched. For codeword distribution along proper decoding paths, the problem is formatted as solving a system of functional equations including four constraints [29]. It is affirmed that rate $R = 0.5$ is a watershed: when $R > 0.5$, DAC codeword distribution is an unsmooth function; while when $R \leq 0.5$, DAC codeword distribution is a smooth function. Especially, a closed form is obtained at $R = 0.5$. In spite of these achievements, it remains a very difficult task to find the closed form of codeword distribution along proper decoding paths in general. As for codeword distribution along wrong decoding paths, only some simulation results are reported in [29], while problem formulation remains an open issue. It deserves to point out that the concept of codeword distribution can be easily extended to the ECAC.

This paper makes some advances on the work in [29]. Three approximation methods are proposed for codeword distribution of equiprobable binary sources along proper decoding paths: numeric approximation, polynomial approximation, and Gaussian approximation. Among them,

numeric approximation is a general approach. At low rates ($R \leq 0.5$), polynomial approximation works well. To reduce computational complexity at very low rates, Gaussian approximation is used as an alternative to polynomial approximation.

This paper is arranged as follows. In Section II, after a brief introduction to binary DAC codec, DAC decoding process is analyzed in detail to show the significance of DAC codeword distribution. Then the investigated problem is formulated in Section III. Section IV, Section V, and Section VI describe in detail numeric approximation, polynomial approximation, and Gaussian approximation, respectively, where simulation results are also reported. Finally, Section VII concludes this paper.

II. BINARY DISTRIBUTED ARITHMETIC CODING

A. Encoding

Consider a binary source $X = \{x_i\}_{i=1}^I$ with bias probability $p = \Pr(x_i = 1)$. In the classic AC, source symbol x_i is iteratively mapped onto sub-intervals of $[0, 1)$, whose lengths are proportional to $(1-p)$ and p , giving rate $R = H(X)$. Instead, in the DAC [22], [23], sub-interval lengths are proportional to enlarged probabilities $(1-p)^\gamma$ and p^γ , where $H(X|Y)/H(X) \leq \gamma \leq 1$, giving rate $R = \gamma H(X) \geq H(X|Y)$. For conciseness, we refer to γ as overlap coefficient hereinafter. More specifically, symbols $x_i = 0$ and $x_i = 1$ correspond to sub-intervals $[0, (1-p)^\gamma)$ and $[1-p^\gamma, 1)$, respectively. It means that to fit the $[0, 1)$ interval, the sub-intervals have to be partially overlapped. This overlapping leads to a larger final interval, and hence a shorter codeword. However, as a cost, the decoder can not decode X unambiguously without Y .

Note that when $\gamma \geq 1/C \geq 1$, where C is channel capacity, it becomes the ECAC.

B. Decoding

To describe the decoding process, a ternary symbol set $\{0, \mathcal{A}, 1\}$ is defined, where \mathcal{A} represents the ambiguous symbol. Let C_X be DAC codeword and \tilde{x}_i be the i -th decoded symbol, then

$$\tilde{x}_i = \begin{cases} 0, & 0 \leq C_X < 1 - p^\gamma \\ \mathcal{A}, & 1 - p^\gamma \leq C_X < (1 - p)^\gamma \\ 1, & (1 - p)^\gamma \leq C_X < 1 \end{cases} \quad (1)$$

After \tilde{x}_i is decoded, if $\tilde{x}_i = \mathcal{A}$, the decoder will perform a branching: two candidate branches are generated, corresponding to two alternative symbols $x_i = 0$ and $x_i = 1$. For each new

branch, its metric is updated and the corresponding interval is selected for next iteration. To reduce complexity, every time a symbol is decoded, the decoder uses the M -algorithm to keep at most M paths with the best partial metric, and prunes others [22], [23]. Finally, after all source symbols are decoded, the path with the best metric is output as the estimate of X . As for detailed performance comparisons between DAC and LDPC-based SWC, please refer to [23].

C. Discussion

It deserves to point out that during DAC decoding, the metric of each path is indeed the Hamming distance between this path and SI Y . As we know, each DAC codeword defines a set of possible decoding paths and each possible decoding path corresponds to a sequence of decoded symbols. However, among all possible decoding paths, there is one and only one proper path which corresponds to source X . Let $\tilde{X} = \{\tilde{x}_i\}_{i=1}^I$ be a sequence of decoded symbols. Let $D(Y, \tilde{X})$ be the Hamming distance between Y and \tilde{X} . Similarly, $D(X, \tilde{X})$ and $D(X, Y)$ are also defined. Obviously,

$$D(Y, \tilde{X}) \leq D(X, Y) + D(X, \tilde{X}). \quad (2)$$

The task of a DAC decoder is in fact to find a path X' that minimizes $D(Y, \tilde{X})$, i.e.

$$X' = \arg \min_{\tilde{X}} D(Y, \tilde{X}). \quad (3)$$

However, this is not always followed by $D(X, X') = 0$. If $D(X, X') \neq 0$, then a decoding failure occurs. To find the probability of decoding failure, we need to know the distribution of $D(Y, \tilde{X})$ and $D(X, \tilde{X})$.

Though it is very difficult to find the distribution of $D(Y, \tilde{X})$ and $D(X, \tilde{X})$, this problem can be tackled by means of DAC codeword distribution. As shown in [29], if we know codeword distributions along proper and wrong decoding paths, it seems promising to find the number of possible decoding paths and the distribution of $D(Y, \tilde{X})$ and $D(X, \tilde{X})$.

The rest of this paper makes some advances on DAC codeword distribution along proper decoding paths.

III. PROBLEM FORMULATION

To simplify the analysis, we consider an infinite-length, stationary, and equiprobable binary source $X = \{x_i\}_{i=1}^{\infty}$. As $p = 0.5$, symbols $x_i = 0$ and $x_i = 1$ correspond to sub-intervals $[0, q)$ and $[1 - q, 1)$ respectively, where $q = 0.5^\gamma$. The resulting rate $R = \gamma H(X) = \gamma$.

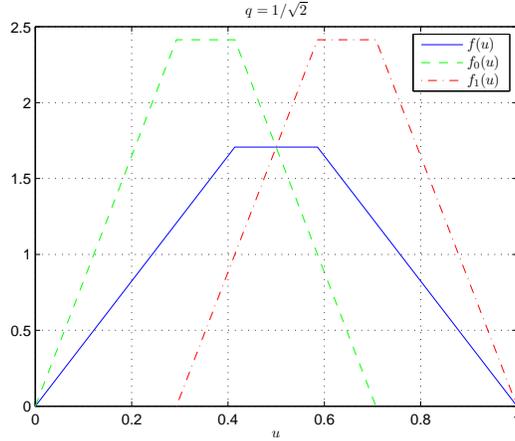


Fig. 1. Illustrations of $f(u)$, $f_0(u)$, and $f_1(u)$ for $q = 1/\sqrt{2}$. $f(u)$ is symmetric around $u = 0.5$, i.e. $f(u) = f(1 - u)$. $f(u)$, $f_0(u)$, and $f_1(u)$ have the same shape. $f(u) = (f_0(u) + f_1(u))/2$. $f_0(u)$ can be obtained by first squeezing $f(u)$ by q times along x -axis and then stretching $f(u)$ by $1/q$ times along y -axis, i.e. $f_0(u) = f(u/q)/q$. $f_1(u)$ can be obtained by shifting $f_0(u)$ right by $(1 - q)$, i.e. $f_1(u) = f_0(u - (1 - q))$. Due to the symmetry, $f_1(u) = f_0(1 - u)$. $f(u)$, $f_0(u)$, and $f_1(u)$ intersect at $u = 0.5$, i.e. $f(0.5) = f_0(0.5) = f_1(0.5)$. Hence $qf(0.5) = f(0.5/q)$.

Let C_X be the DAC codeword of X and $f(u)$ ($0 \leq u < 1$) be the distribution of C_X , then

$$\int_0^1 f(u) du = 1. \quad (4)$$

Due to the symmetry, we have

$$f(u) = f(1 - u), \quad 0 < u < 1. \quad (5)$$

Symbols $x_1 = 0$ and $x_1 = 1$ correspond to intervals $[0, q)$ and $[1 - q, 1)$, respectively. If $x_1 = 0$, the remaining sequence $X_2 = \{x_i\}_{i=2}^{\infty}$ will be iteratively mapped onto the sub-intervals of $[0, q)$; otherwise, X_2 will be iteratively mapped onto the sub-intervals of $[1 - q, 1)$. Let $C_{X_2}^0$ be the DAC codeword of X_2 given $x_1 = 0$ and $f_0(u)$ be the distribution of $C_{X_2}^0$, then

$$\int_0^q f_0(u) du = 1. \quad (6)$$

Since X is infinite-length and stationary, $f_0(u)$ must have the same shape as $f(u)$, i.e.,

$$f_0(u) = f(u/q)/q, \quad 0 \leq u < q. \quad (7)$$

Similarly, let $C_{X_2}^1$ be the DAC codeword of X_2 given $x_1 = 1$ and $f_1(u)$ be the distribution of $C_{X_2}^1$, then

$$f_1(u) = f_0(u - (1 - q)) = f\left(\frac{u - (1 - q)}{q}\right)/q, \quad (1 - q) \leq u < 1. \quad (8)$$

Due to the symmetry,

$$f_1(u) = f_0(1 - u). \quad (9)$$

The relations between $f(u)$, $f_0(u)$, and $f_1(u)$ can be illustrated by Fig.1. Obviously,

$$f(u) = \Pr(x_1 = 0)f_0(u) + \Pr(x_1 = 1)f_1(u) = (f_0(u) + f_1(u))/2. \quad (10)$$

Hence, $f(u)$, $f_0(u)$, and $f_1(u)$ intersect at $u = 0.5$, i.e. $f(0.5) = f_0(0.5) = f_1(0.5)$. Thus

$$qf(0.5) = f(0.5/q). \quad (11)$$

A. Classic AC

When $q = 0.5$, it is just the classic AC. Then

$$f(u) = \begin{cases} f(2u), & 0 \leq u < 0.5 \\ f(2u - 1), & 0.5 \leq u < 1 \end{cases}. \quad (12)$$

It is easy to prove $f(u) \equiv 1$ ($0 \leq u < 1$). This is a uniform distribution, so the classic AC can achieve source entropy theoretically.

B. Distributed AC

When $0.5 < q < 1$, sub-intervals $[0, q)$ and $[1 - q, 1)$ are partially overlapped, so $f(u)$ is a piecewise-defined function.

1) $0 \leq u < (1 - q)$: In this interval, $f_1(u) = 0$, so

$$f(u) = f_0(u)/2 = f(u/q)/(2q). \quad (13)$$

Since $f(0) = f(0/q)/(2q)$, we have $f(0) = 0$.

2) $q \leq u < 1$: In this interval, $f_0(u) = 0$, so

$$f(u) = f_1(u)/2 = f\left(\frac{u - (1 - q)}{q}\right)/(2q). \quad (14)$$

3) $1 - q \leq u < q$: In this interval, we have

$$f(u) = \frac{f\left(\frac{u}{q}\right) + f\left(\frac{u - (1 - q)}{q}\right)}{2q}. \quad (15)$$

C. A Closed Form of $f(u)$ at $q = 1/\sqrt{2}$

Generally, it is very difficult to obtain the closed form of $f(u)$. In [29], only one closed form is obtained at $q = 1/\sqrt{2}$ (i.e. $\gamma = 0.5$):

$$f(u) = \begin{cases} \frac{u}{3\sqrt{2}-4}, & 0 \leq u \leq \sqrt{2}-1 \\ \frac{1}{2-\sqrt{2}}, & \sqrt{2}-1 \leq u \leq 2-\sqrt{2} \\ \frac{1-u}{3\sqrt{2}-4}, & 2-\sqrt{2} \leq u \leq 1 \end{cases} \quad (16)$$

D. Zeros of $f(u)$ at High Rates

It is proved in [29] that when $0.5 < q \leq \frac{\sqrt{5}-1}{2}$ (corresponds to $0.6942 \leq \gamma < 1$), $f(\frac{q^n}{q+1}) = f(1 - \frac{q^n}{q+1}) = 0, \forall n \in \mathbb{N}$.

IV. NUMERIC APPROXIMATION

Though a special closed form of $f(u)$ is found for $p = \gamma = 0.5$ in [29], the procedure is very complex. In general, the closed form of $f(u)$ does not exist. As a universal approach, we propose a numeric method for finding $f(u)$. This method is described in detail below.

A. Discretization

We divide the interval $[0, 1]$ into N uniform cells. Let $\Delta = 1/N$. Then $f(u)$ can be approximated by $f(n\Delta)$, where $n \in \mathcal{I}_N = \{0, 1, \dots, N\}$, given a large N .

B. Initialization

Let $f^{(t)}(n\Delta)$ be the estimate of $f(n\Delta)$ after t iterations. Before iteration, $f^{(0)}(n\Delta)$ need to be initialized. Though arbitrary initialization is allowed, we recommend uniform initialization, i.e. $f^{(0)}(n\Delta) \equiv 1$, where $n \in \mathcal{I}_N$.

C. Iteration

Let $L = \lfloor N(1-q) \rfloor = N - \lceil Nq \rceil$ and $H = \lceil Nq \rceil$. Then the iteration is run as follows.

1) $0 \leq n \leq L$: This corresponds to interval $0 \leq u \leq (1 - q)$, hence

$$f^{(t)}(n\Delta) = \frac{f^{(t-1)}(h_{0,N}(n/q)\Delta)}{2q}, \quad (17)$$

where

$$h_{a,b}(x) = \begin{cases} a, & \text{round}(x) < a < b \\ \text{round}(x), & a \leq \text{round}(x) \leq b. \\ b, & a < b < \text{round}(x) \end{cases} \quad (18)$$

2) $H \leq n \leq N$: This corresponds to interval $q \leq u \leq 1$. Because $L + H = N$,

$$f^{(t)}(n\Delta) = f^{(t)}((N - n)\Delta). \quad (19)$$

3) $L < n < H$: This corresponds to interval $(1 - q) < u < q$, hence

$$f^{(t)}(n\Delta) = \frac{f^{(t-1)}(h_{0,N}(n/q)\Delta) + f^{(t-1)}(h_{0,N}(\frac{n-L}{q})\Delta)}{2q}. \quad (20)$$

D. Normalization

Recall the constraint $\int_0^1 f(u)du = 1$, we have

$$\sum_{n=0}^N f^{(t)}(n\Delta)\Delta = 1, \quad (21)$$

i.e.

$$\sum_{n=0}^N f^{(t)}(n\Delta) = 1/\Delta = N. \quad (22)$$

Let $\sum_{n=0}^N f^{(t)}(n\Delta) = \Omega$, then $f^{(t)}(n\Delta)$ should be normalized as below:

$$f^{(t)}(n\Delta) = \frac{Nf^{(t)}(n\Delta)}{\Omega}. \quad (23)$$

E. Termination

We use the Mean Squared Error (MSE) between two successive iterations as a measurement to terminate the iteration. Let δ be a small quantity. When

$$\text{MSE}^{(t)} = \frac{1}{N+1} \sum_{n=0}^N (f^{(t)}(n\Delta) - f^{(t-1)}(n\Delta))^2 < \delta, \quad (24)$$

the iteration is terminated.

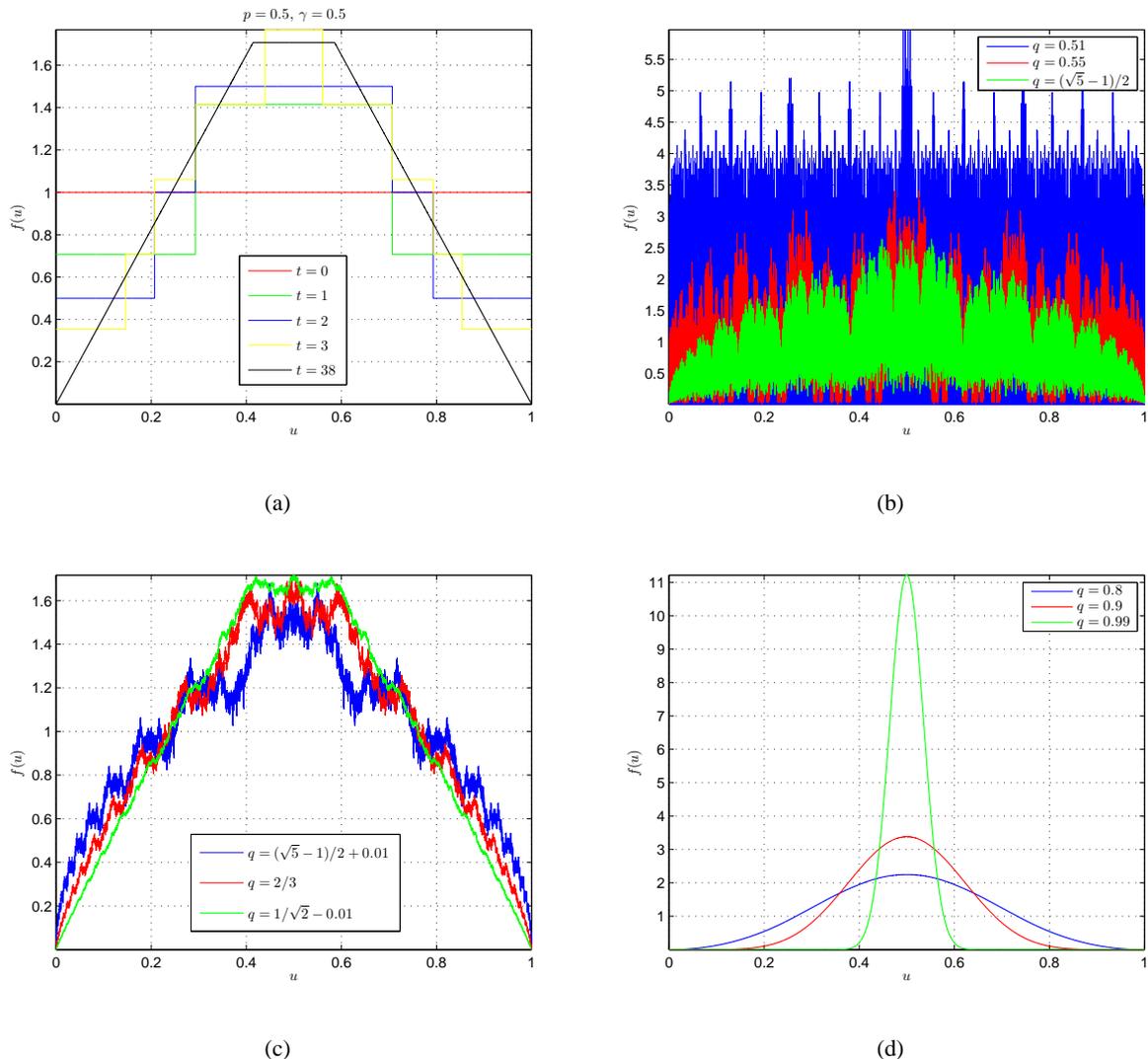


Fig. 2. Simulation curves of numeric approximation to $f(u)$, where $N = 10^5$. All these results coincides with those given in [29], meaning that numeric approximation is well justified. (a) Evolution of $f^{(t)}(n\Delta)$ with respect to t for $q = 1/\sqrt{2}$. As t increases, $f^{(t)}(n\Delta)$ converges to $f(u)$. $\text{MSE}^{(38)} < 10^{-10}$. (b) Some results for $q \in (0.5, (\sqrt{5}-1)/2]$. $\text{MSE}^{(586)} < 10^{-4}$ for $q = 0.51$. $\text{MSE}^{(70)} < 10^{-4}$ for $q = 0.55$. $\text{MSE}^{(51)} < 10^{-4}$ for $q = (\sqrt{5}-1)/2$. (c) Some results for $q \in ((\sqrt{5}-1)/2, 1/\sqrt{2})$. $\text{MSE}^{(85)} < 10^{-9}$ for $q = (\sqrt{5}-1)/2 + 0.01$. $\text{MSE}^{(63)} < 10^{-9}$ for $q = 2/3$. $\text{MSE}^{(52)} < 10^{-9}$ for $q = 1/\sqrt{2} - 0.01$. (d) Some results for $q \in [1/\sqrt{2}, 1)$. $\text{MSE}^{(39)} < 10^{-10}$ for $q = 0.8$. $\text{MSE}^{(54)} < 10^{-10}$ for $q = 0.9$. $\text{MSE}^{(540)} < 10^{-9}$ for $q = 0.99$.

F. Simulation Results

Fig. 2 includes some results regarding numeric approximation. All results reported in Fig. 2 are obtained with $N = 10^5$.

To show how $f^{(t)}(n\Delta)$ converges to $f(u)$, the evolution of $f^{(t)}(n\Delta)$ with t is plotted in Fig. 2(a). We find that after 38 iterations, successive MSE has been less than 10^{-10} .

It was affirmed in [29] that $(\sqrt{5} - 1)/2$ and $1/\sqrt{2}$ are two watersheds that divide interval $(0.5, 1)$ of q into three sub-intervals: $(0.5, (\sqrt{5} - 1)/2]$, $((\sqrt{5} - 1)/2, 1/\sqrt{2})$, and $[1/\sqrt{2}, 1)$, because $f(u)$ shows very different properties in these three sub-intervals. As in [29], for each sub-interval of q , some simulation results are reported in Figs. 2(b)-(d). All these results coincide with those given in [29] perfectly. Fig. 2(b) confirms the zeros of $f(u)$ at high rates. Fig. 2(d) shows that $f(u)$ becomes smooth at low rates.

In different sub-intervals, numeric approximation shows very different simulation precision and computational complexity. Firstly, we consider simulation precision. For $q \in (0.5, (\sqrt{5} - 1)/2]$, tens of iterations are needed to make successive MSE less than 10^{-4} , while for $q \in ((\sqrt{5} - 1)/2, 1/\sqrt{2})$, tens of iterations have made successive MSE less than 10^{-9} . For $q \in [1/\sqrt{2}, 1)$, tens of iterations can even make successive MSE less than 10^{-10} . Secondly, we consider computational complexity. We find that $q = 1/\sqrt{2}$ needs the fewest iterations. As q departs from $1/\sqrt{2}$ (increase or decrease), computational complexity increases, i.e. more iterations are needed to reach the same successive MSE. Thirdly, as q approaches to 0.5 or 1, simulation precision is sharply degraded, or in other words, computational complexity increases sharply. For example, when $q = 0.51$, 586 iterations are needed to make successive MSE less than 10^{-4} , while for other q in the same sub-intervals (e.g. 0.55 and $(\sqrt{5} - 1)/2$), tens of iterations are enough to reach the same precision. Similar phenomenon is also observed for $q = 0.99$.

It may be an interesting issue to improve simulation precision and accelerate convergence speed of $f(u)$, especially for q close to 0.5 or 1.

V. POLYNOMIAL APPROXIMATION AT LOW RATES

It was affirmed in [29] that $f(u)$ is a smooth function when $q \geq 1/\sqrt{2}$, i.e. $R \leq 0.5$. This property suggests that polynomials may be good approximation to $f(u)$ at low rates ($R \leq 0.5$). Below we propose polynomial approximation to $f(u)$ for $1/\sqrt{2} \leq q < 1$.

To simplify the analysis, we exploit the symmetry and consider only the left half of $f(u)$

$$f(u) = \begin{cases} f(u/q)/(2q), & 0 \leq u \leq (1-q) \\ \frac{f(\frac{u}{q}) + f(\frac{u-(1-q)}{q})}{2q}, & (1-q) \leq u \leq 0.5 \end{cases}. \quad (25)$$

We rewrite (25) as

$$f(u) = \begin{cases} 2qf(qu), & 0 \leq u \leq v_1 \\ 2qf(qu) - f(u - v_1), & v_1 \leq u \leq 0.5 \end{cases}. \quad (26)$$

where $v_n = (1-q)/q^n$, $n \in \mathbb{N}$. Note that $v_1 < 0.5$ when $q \geq 1/\sqrt{2}$. Hence, $f(u)$ is a piecewise-defined function over interval $[0, 0.5]$.

At first, in sub-interval $[0, v_1]$, $f(u)$ can be obtained by solving functional equation $f(u) = 2qf(qu)$ [30]

$$f(u) = \phi(u) = \Theta(u)u^\lambda, \quad 0 \leq u \leq v_1, \quad (27)$$

where $\lambda = (1-\gamma)/\gamma$ and $\Theta(u) = \Theta(uq^k)$, $\forall k \in \mathbb{Z}$.

Then, we need to determine $f(u)$ in sub-interval $[v_1, 0.5]$. Because $qu < u$ and $u - v_1 < u$, it is possible to recursively map sub-interval $[v_1, 0.5]$ onto sub-interval $[0, v_1]$ by scaling down or shifting u , over which $f(u)$ has been given by (27), i.e.

$$\begin{cases} f(qu) = \phi(qu), & v_1 \leq u \leq v_2 \\ f(u - v_1) = \phi(u - v_1), & v_1 \leq u \leq 2v_1 \end{cases}. \quad (28)$$

This is the key to solving this problem.

It is easy to prove that $u - v_1 < qu$ for $u \in [v_1, 0.5]$. Hence,

$$f(u) = 2qf(qu) - \phi(u - v_1), \quad v_1 \leq u \leq 2v_1. \quad (29)$$

On solving $2v_1 = 0.5$, we obtain $q = 0.8$. Hereinafter, to facilitate our description, we divide interval $1/\sqrt{2} \leq q < 1$ into two sub-intervals $1/\sqrt{2} \leq q \leq 0.8$ (corresponding to $0.5 \leq 2v_1$) and $0.8 < q < 1$ (corresponding to $2v_1 < 0.5$).

A. $1/\sqrt{2} \leq q \leq 0.8$

In this sub-interval, since $0.5 \leq 2v_1$, we have

$$f(u) = \begin{cases} \phi(u), & 0 \leq u \leq v_1 \\ 2qf(qu) - \phi(u - v_1), & v_1 \leq u \leq 0.5 \end{cases}. \quad (30)$$

Hence, we need to consider only the term $2qf(qu)$. Depending on the relations between v_n and 0.5, this sub-interval can be further divided into three smaller sub-intervals.

1) $0.5 \leq v_2$: On solving $v_2 = 0.5$, we obtain $q = \sqrt{3} - 1$, so this sub-interval corresponds to $1/\sqrt{2} \leq q \leq \sqrt{3} - 1$. Since $0.5 \leq v_2$, we have $qu \leq v_1$ for $u \in [v_1, 0.5]$, i.e. $f(qu) = \phi(qu)$. Remember $\phi(u) \equiv 2q\phi(qu)$. Thus

$$f(u) = \begin{cases} \phi(u), & 0 \leq u \leq v_1 \\ \phi(u) - \phi(u - v_1), & v_1 \leq u \leq 0.5 \end{cases}. \quad (31)$$

As affirmed in [29], $f(u)$ is a smooth function for $q \geq 1/\sqrt{2}$. Hence we approximate $\Theta(u)$ by a const c and then obtain

$$f(u) \approx \begin{cases} cu^\lambda, & 0 \leq u \leq v_1 \\ cu^\lambda - c(u - v_1)^\lambda, & v_1 \leq u \leq 0.5 \end{cases}. \quad (32)$$

Now we need to determine c . Let us integrate $f(u)$ over interval $[0, 0.5]$

$$\begin{aligned} \int_0^{0.5} f(u)du &= c \left(\int_0^{0.5} u^\lambda du - \int_{v_1}^{0.5} (u - v_1)^\lambda du \right) \\ &= \frac{c(u^{\lambda+1}|_0^{0.5} - (u - v_1)^{\lambda+1}|_{v_1}^{0.5})}{\lambda + 1} \\ &= \frac{c(0.5^{\lambda+1} - (0.5 - v_1)^{\lambda+1})}{\lambda + 1} = 0.5. \end{aligned} \quad (33)$$

Thus,

$$c = \frac{0.5(\lambda + 1)}{0.5^{\lambda+1} - (0.5 - v_1)^{\lambda+1}}. \quad (34)$$

Due to $\lambda + 1 = 1/\gamma$,

$$c = \frac{1}{2\gamma(0.5^{(1/\gamma)} - (0.5 - v_1)^{(1/\gamma)})}. \quad (35)$$

2) $v_2 < 0.5 \leq v_3$: On solving $v_3 = 0.5$, we obtain $q \approx 0.77$, so this sub-interval corresponds to $\sqrt{3} - 1 < q \leq 0.77$. At first, it can be obtained directly

$$f(u) = \begin{cases} \phi(u), & 0 \leq u \leq v_1 \\ \phi(u) - \phi(u - v_1), & v_1 \leq u \leq v_2 \end{cases}. \quad (36)$$

Then, for $u \in [v_2, 0.5]$, we have $qu \in [v_1, v_2]$, i.e. $f(qu) = \phi(qu) - \phi(qu - v_1)$. Thus

$$\begin{aligned} f(u) &= 2qf(qu) - \phi(u - v_1) \\ &= 2q(\phi(qu) - \phi(qu - v_1)) - \phi(u - v_1), \quad v_2 \leq u \leq 0.5. \end{aligned} \quad (37)$$

Because $2q\phi(qu - v_1) = 2q\phi(q(u - v_2)) = \phi(u - v_2)$, we obtain

$$f(u) = \phi(u) - \sum_{i=1}^2 \phi(u - v_i), \quad v_2 \leq u \leq 0.5. \quad (38)$$

Therefore, we can obtain the following approximation

$$f(u) \approx \begin{cases} cu^\lambda, & 0 \leq u \leq v_1 \\ cu^\lambda - c(u - v_1)^\lambda, & v_1 \leq u \leq v_2, \\ cu^\lambda - c \sum_{i=1}^2 (u - v_i)^\lambda, & v_2 \leq u \leq 0.5 \end{cases}, \quad (39)$$

where

$$c = \frac{1}{2\gamma(0.5^{(1/\gamma)} - \sum_{i=1}^2 (0.5 - v_i)^{(1/\gamma)})}. \quad (40)$$

3) $v_3 < 0.5 \leq v_4$: On solving $v_4 = 0.5$, we obtain $q \approx 0.8$, so this sub-interval corresponds to $0.77 < q \leq 0.8$. By iterations, we can obtain

$$f(u) \approx \begin{cases} cu^\lambda, & 0 \leq u \leq v_1 \\ cu^\lambda - c(u - v_1)^\lambda, & v_1 \leq u \leq v_2 \\ cu^\lambda - c \sum_{i=1}^2 (u - v_i)^\lambda, & v_2 \leq u \leq v_3, \\ cu^\lambda - c \sum_{i=1}^3 (u - v_i)^\lambda, & v_3 \leq u \leq 0.5 \end{cases}, \quad (41)$$

where

$$c = \frac{1}{2\gamma(0.5^{(1/\gamma)} - \sum_{i=1}^3 (0.5 - v_i)^{(1/\gamma)})}. \quad (42)$$

B. $0.8 < q < 1$

The problem becomes very complex in this sub-interval because $f(u - v_1) = \phi(u - v_1)$ does not hold for $u \in [2v_1, 0.5]$ so that we need to deal with not only $2qf(qu)$ but also $f(u - v_1)$.

Let us consider a simple case first, i.e. $v_1 < 0.5 - v_1 \leq v_2$, which corresponds to sub-interval $0.8 < q \leq \sqrt{2/3}$. We have $u - v_1 \in [v_1, v_2]$ for $u \in [2v_1, 0.5]$. Hence

$$f(u - v_1) = \phi(u - v_1) - \phi(u - 2v_1), \quad 2v_1 \leq u \leq 0.5. \quad (43)$$

Therefore, the problem becomes

$$f(u) = \begin{cases} 2qf(qu), & 0 \leq u \leq v_1 \\ 2qf(qu) - \phi(u - v_1), & v_1 \leq u \leq 2v_1. \\ 2qf(qu) - (\phi(u - v_1) - \phi(u - 2v_1)), & 2v_1 \leq u \leq 0.5 \end{cases}. \quad (44)$$

Now we need to deal with only $2qf(qu)$, which has been discussed in detail in Section V-A.

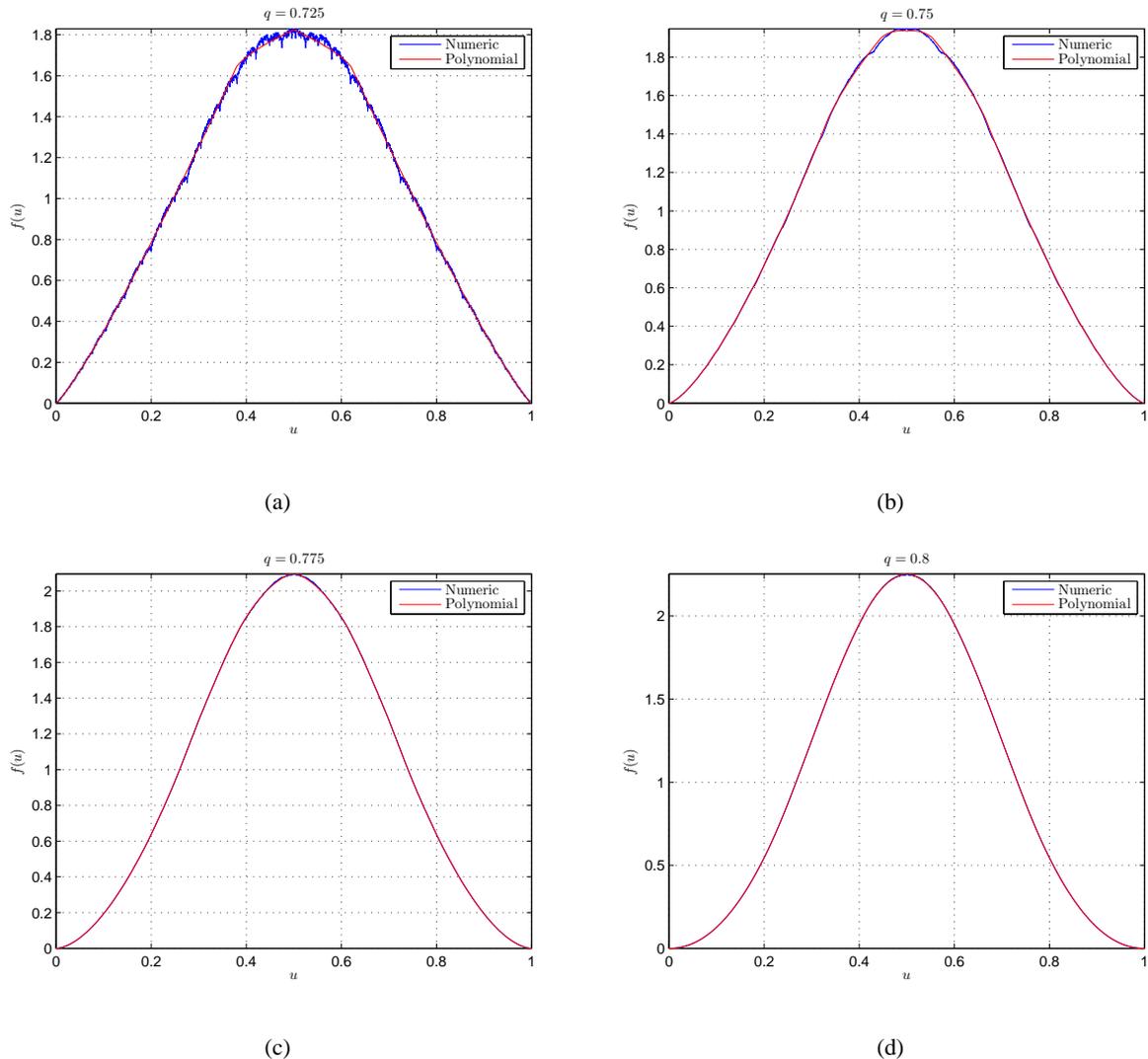


Fig. 3. Comparisons of polynomial approximation with numeric approximation, where $N = 10^5$ and $\delta = 10^{-10}$ for numeric approximation. These results show that polynomial approximation fits numeric approximation very well. Especially, as q increases, polynomial approximation almost coincides with numeric approximation. (a) $q = 0.725$. (b) $q = 0.75$. (c) $q = 0.775$. (d) $q = 0.8$.

For $\sqrt{2/3} < q < 1$, the idea is the same but the procedure becomes more and more complicated as q increases. Therefore, at very low rates, polynomial approximation is not a good choice.

C. Simulation Results

Some examples of polynomial approximation have been included in Fig. 3. Considering the complexity, only the results for $1/\sqrt{2} \leq q \leq 0.8$ are reported. Fig. 3 shows that in general, the

curves of polynomial approximation fit those of numeric approximation very well. Especially, as q increases, the curves of polynomial approximation almost coincide with those of numeric approximation. In addition, Fig. 3(a) also shows the affirmation in [29] may fail because $q > 1/\sqrt{2}$ does not guarantee smooth $f(u)$. Nevertheless, $f(u)$ does become less irregular as q increases.

VI. GAUSSIAN APPROXIMATION AT VERY LOW RATES

As pointed out in Section V that as q increases, polynomial approximation to $f(u)$ becomes very complex. Thus a simpler approximation method is needed at very low rates. Through experiments, we observe that $f(u)$ becomes bell-shaped at very low rates [29]. This phenomenon suggests that a Gaussian function centered at 0.5 may be good approximation to $f(u)$, i.e.

$$f(u) \approx \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(u-0.5)^2}{2\sigma^2}\right). \quad (45)$$

Obviously, the problem now boils down to how to estimate σ^2 for given q .

A. Estimation of σ^2

Here we propose a simple method to estimate σ^2 by exploiting $qf(0.5) = f(0.5/q)$ [Fig. 1]. For a large q , we have

$$qf(0.5) \approx \frac{q}{\sqrt{2\pi}\sigma} \quad (46)$$

and

$$f(0.5/q) \approx \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(1-q)^2}{8q^2\sigma^2}\right). \quad (47)$$

Hence,

$$q \approx \exp\left(-\frac{(1-q)^2}{8q^2\sigma^2}\right). \quad (48)$$

Therefore

$$\sigma^2 \approx -\frac{(1-q)^2}{8q^2 \ln q}. \quad (49)$$

B. Simulation Results

Some examples of Gaussian approximation are included in Fig. 4. These plots show that as q increases, the curves of Gaussian approximation become closer and closer to those of numeric approximation. Especially, when $q = 0.99$, the curve of Gaussian approximation almost coincides with that of numeric approximation. All these results confirm that Gaussian approximation does work well at very low rates.

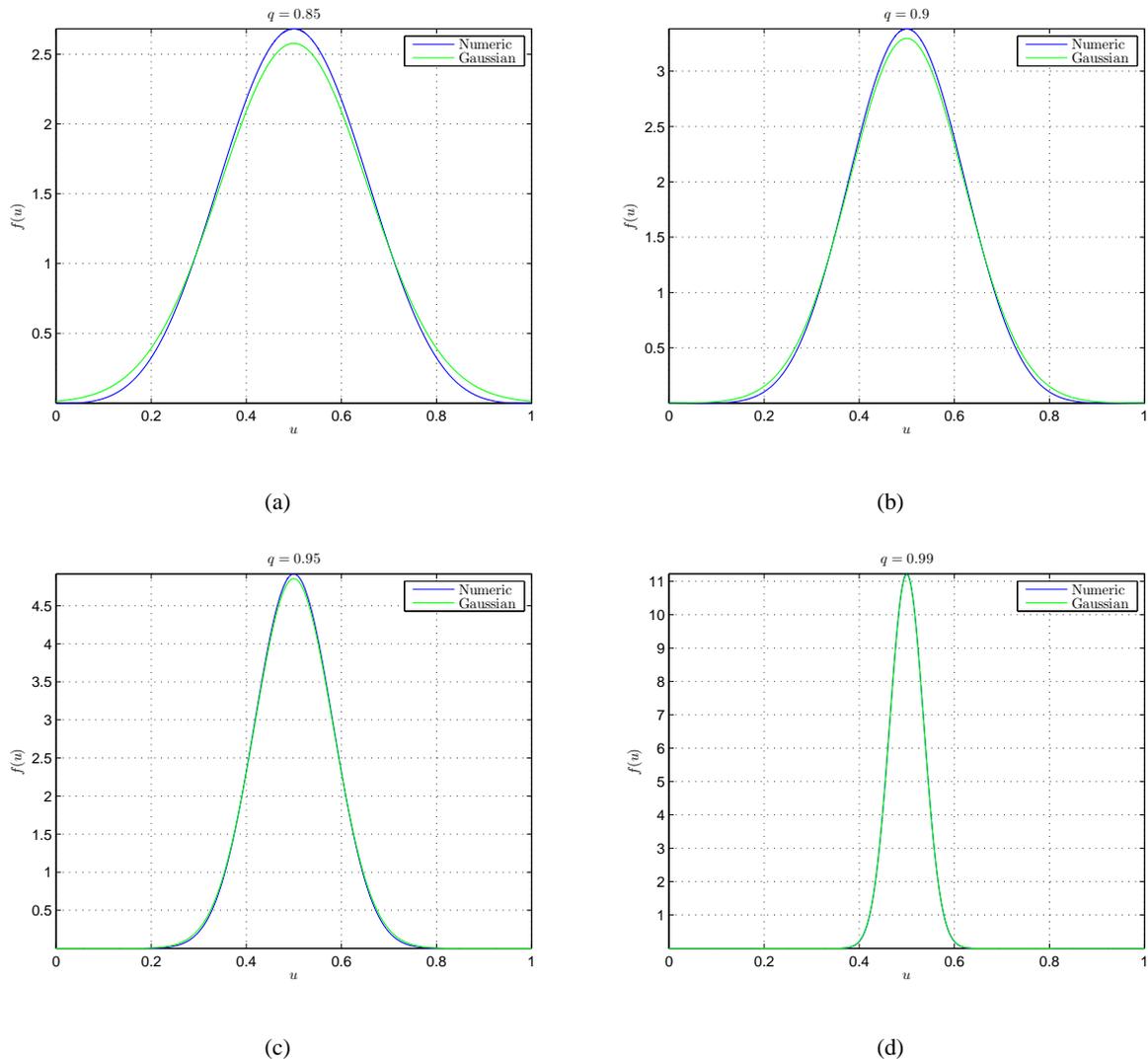


Fig. 4. Comparisons of Gaussian approximation with numeric approximation, where $N = 10^5$ for numeric approximation. As q increases, Gaussian approximation becomes more and more accurate. (a) $q = 0.85$. $\delta = 10^{-10}$ for numeric approximation. (b) $q = 0.9$. $\delta = 10^{-10}$ for numeric approximation. (c) $q = 0.95$. $\delta = 10^{-10}$ for numeric approximation. (d) $q = 0.99$. $\delta = 10^{-9}$ for numeric approximation.

VII. CONCLUSION

This paper proposes three approximation methods for DAC codeword distribution of equiprobable binary sources along proper decoding paths. These methods are well justified by simulation results. The related software is available on [31].

Nevertheless, there remain many open issues. Firstly, how to format the problem for codeword

distribution along wrong decoding path? Secondly, for general (non-equiprobable or M -ary) sources, how to format the problem? Thirdly, can we find the number of possible decoding paths as well as the distributions of $D(X, \tilde{X})$ and $D(Y, \tilde{X})$, for a given DAC code of X . Finally, it is an interesting issue to define codeword distribution for the ECAC.

REFERENCES

- [1] D. Slepian and J. K. Wolf, "Noiseless coding of correlated information sources," *IEEE Trans. Inf. Theory*, vol. 19, no. 4, pp. 471–480, July 1973.
- [2] J. Garcia-Frias and Y. Zhao, "Compression of correlated binary sources using turbo codes," *IEEE Commun. Lett.*, vol. 5, no. 10, pp. 417–419, Oct. 2001.
- [3] A. Liveris, Z. Xiong, and C. Georghiades, "Compression of binary sources with side information at the decoder using LDPC codes," *IEEE Commun. Lett.*, vol. 6, no. 10, pp. 440–442, Oct. 2002.
- [4] J. Rissanen, "Generalized Kraft inequality and arithmetic coding," *IBM J. Research and Development*, vol. 20, no. 3, pp. 198–203, May 1976.
- [5] J. Rissanen and G. Langdon, "Arithmetic coding," *IBM J. Research and Development*, vol. 23, no. 2, pp. 149–162, Mar. 1979.
- [6] J. J. Rissanen, "Arithmetic codings as number representations," *Acta Polytechnica Scandinavica*, vol. 31, pp. 44–51, Dec. 1979.
- [7] P. G. Howard and J. S. Vitter, "Practical implementations of arithmetic coding," in *Image and Text Compression*, pp. 85–112, Kluwer Academic, Norwell, Mass, USA, 1992.
- [8] G. F. Elmasry, "Embedding channel coding in arithmetic coding," *IEE Proc. Commun.*, vol. 146, no. 2, pp. 73–78, Feb. 1999.
- [9] I. Sodagar, B. B. Chai, and J. Wus, "A new error resilience technique for image compression using arithmetic coding," in *Proc. IEEE ICASSP*, pp. 2127–2130, Istanbul, Turkey, June 2000.
- [10] C. Boyd, J. G. Cleary, S. A. Irvine, I. Rinsma-Melchert, and I.H. Witten, "Integrating error detection into arithmetic coding," *IEEE Trans. Commun.*, vol. 45, no. 1, pp. 1–3, Jan. 1997.
- [11] G. F. Elmasry, "Joint lossless-source and channel coding using automatic repeat request," *IEEE Trans. Commun.*, vol. 47, no. 7, pp. 953–955, Jul. 1999.
- [12] J. Chou and K. Ramchandran, "Arithmetic coding-based continuous error detection for efficient ARQ-based image transmission," *IEEE J. Selected Areas in Commun.*, vol. 18, no. 6, pp. 861–867, June 2000.
- [13] R. Anand, K. Ramchandran, and I. Kozintsev, "Continuous error detection (CED) for reliable communication," *IEEE Trans. Commun.*, vol. 49, no. 9, pp. 1540–1549, Sep. 2001.
- [14] B.D. Pettijohn, M.W. Hoffman, and K. Sayood, "Joint source/channel coding using arithmetic codes", *IEEE Trans. Commun.*, vol. 49, no. 5, pp. 826–836, May 2001.
- [15] C. Demiroglu, M. W. Hoffman, and K. Sayood, "Joint source/channel coding using arithmetic codes and trellis coded modulation," in *Proc. IEEE DCC*, pp. 302–311, Snowbird, Utah, USA, Mar. 2001
- [16] T. Guionnet and C. Guillemot, "Soft decoding and synchronization of arithmetic codes: application to image transmission over noisy channels," *IEEE Trans. Image Process.*, vol. 12, no. 12, pp. 1599–1609, Dec. 2003.

- [17] T. Guionnet and C. Guillemot, "Soft and joint source-channel decoding of quasi-arithmetic codes," *EURASIP J. Applied Signal Process.*, pp. 393–411, Mar. 2004.
- [18] M. Grangetto, E. Magli, and G. Olmo, "Robust video transmission over error-prone channels via error correcting arithmetic codes," *IEEE Commun. Lett.*, vol. 7, no. 12, pp. 596–598, Dec. 2003.
- [19] M. Grangetto, P. Cosman, and G. Olmo, "Joint source/channel coding and MAP decoding of arithmetic codes," *IEEE Trans. Commun.*, vol. 53, no. 6, pp. 1007–1016, Jun. 2005.
- [20] M. Grangetto, E. Magli, and G. Olmo, "A syntax preserving error resilience tool for JPEG 2000 based on error correcting arithmetic coding," *IEEE Trans. Image Process.*, vol. 15, no. 4, pp. 807–818, Apr. 2006.
- [21] M. Grangetto, B. Scanavino, G. Olmo, and S. Benedetto, "Iterative decoding of serially concatenated arithmetic and channel codes with JPEG 2000 applications," *IEEE Trans. Image Process.*, vol. 16, no. 6, pp. 1557–1567, Jun. 2007.
- [22] M. Grangetto, E. Magli, and G. Olmo, "Distributed arithmetic coding," *IEEE Commun. Lett.*, vol. 11, no. 11, pp. 883–885, Nov. 2007.
- [23] M. Grangetto, E. Magli, and G. Olmo, "Distributed arithmetic coding for the Slepian-Wolf problem," *IEEE Trans. Signal Process.*, vol. 57, no. 6, pp. 2245–2257, Jun. 2009.
- [24] X. Artigas, S. Malinowski, C. Guillemot, and L. Torres, "Overlapped quasi-arithmetic codes for distributed video coding," in *Proc. IEEE ICIP, 2007*, vol. II, pp. 9–12.
- [25] S. Malinowski, X. Artigas, C. Guillemot, and L. Torres, "Distributed coding using punctured quasi-arithmetic codes for memory and memoryless sources," in *Proc. IEEE PCS, Chicago, IL, May 2009*.
- [26] M. Grangetto, E. Magli, and G. Olmo, "Symmetric distributed arithmetic coding of correlated sources," in *Proc. IEEE MMSP, 2007*, pp. 111–114.
- [27] M. Grangetto, E. Magli, R. Tron, and G. Olmo, "Rate-compatible distributed arithmetic coding," *IEEE Commun. Lett.*, vol. 12, no. 8, pp. 575–577, Aug. 2008.
- [28] M. Grangetto, E. Magli, and G. Olmo, "Decoder-driven adaptive distributed arithmetic coding," in *Proc. IEEE ICIP, 2008*, pp. 1128–1131.
- [29] Y. Fang, "Distribution of distributed arithmetic codewords for equiprobable binary sources," *IEEE Signal Process. Lett.*, vol. 16, no. 12, pp. 1079–1082, Dec. 2009.
- [30] <http://eqworld.ipmnet.ru/en/solutions/fe/fe1111.pdf>.
- [31] <http://www.wavesharp.com/fangyong/>.