Probability Interval Partitioning Entropy Codes

Detlev Marpe, *Senior Member, IEEE,* Heiko Schwarz, and Thomas Wiegand, *Senior Member, IEEE*

***Abstract*—A novel approach to entropy coding is described that provides the coding efficiency and simple probability modeling capability of arithmetic coding at the complexity level of Huffman coding. The key element of the proposed approach is given by a partitioning of the unit interval into a small set of disjoint probability intervals for pipelining the coding process along the probability estimates of binary random variables. According to this partitioning, an input sequence of discrete source symbols with arbitrary alphabet sizes is mapped to a sequence of binary symbols and each of the binary symbols is assigned to one particular probability interval. With each of the intervals being represented by a fixed probability, the probability interval partitioning entropy (PIPE) coding process is based on the design and application of simple variable-to-variable length codes. Probability modeling can either be fixed or adaptive while the PIPE code for each probability interval remains fixed and is decoupled from the modeling stage. The excess rate of PIPE coding is shown to be comparable to that of arithmetic coding.**

***Index Terms*—Entropy coding, Huffman coding, arithmetic coding**

I. Introduction

ENTROPY CODING is a reversible mapping from one data representation to another more compact representation. The data can be given in the form of text, graphics, images, video, audio, speech, facsimile, medical data, meteorological data, financial data, or any other form of digital data. In many coding applications and coding standards such as [1]–[6], the original source data are first mapped onto so-called coding symbols and these coding symbols are then entropy coded. The mapping to coding symbols can include quantization, in which case the overall coding scheme is lossy.

It is well known that for a random variable S over an alphabet $\mathcal{A} = \{a_0, \ldots, a_{M-1}\}$, with $M \geq 2$, and an associated probability mass function (pmf) $p_S(a) = P(S = a)$ as defined for each $a \in \mathcal{A}$, the entropy [7]

$$H(S) = -\sum_{m=0}^{M-1} p_S(a_m)\ \log_2 p_S(a_m) \tag{1}$$

is the greatest lower bound for the average codeword length in bits per symbol, for coding the random variable S, that can be achieved with entropy coding techniques. In practice, sources that contain statistical dependencies between samples are processed using conditional probabilities and the probability values are simultaneously estimated at encoder and decoder. This processing is called the probability modeling stage in entropy coding while the assignment of codewords to symbols is the actual entropy coding. For decades, two methods have dominated practical entropy coding: *Huffman coding* that has been invented in 1952 [8] and *arithmetic coding* that goes back to initial ideas attributed to Shannon [7] and Elias [9] and for which first practical schemes have been published around 1976 [10][11]. Both entropy coding methods are capable of approximating the entropy limit (in a certain sense) [12].

For a fixed probability mass function, Huffman codes are relatively easy to construct. The most attractive property of Huffman codes is that their implementation can be efficiently realized by the use of variable-length code (VLC) tables. However, when dealing with time-varying source statistics, i.e., changing symbol probabilities, the adaptation of Huffman codes and its corresponding VLC tables is quite demanding, both in terms of algorithmic complexity as well as in terms of implementation costs. Also, in the case of having a dominant alphabet value with $p_S(a) > 0.5$, the redundancy of the corresponding Huffman code (without using any alphabet extension such as run length coding) may be quite substantial. Another shortcoming of Huffman codes is given by the fact that in case of dealing with higher-order probability modeling, multiple sets of VLC tables may be required.

On the other hand, arithmetic coding, while being substantially more complex than VLC, offers the advantage of a more consistent and adequate handling when coping with adaptive and higher-order probability modeling as well as with the case of highly skewed probability mass functions. Actually, this characteristic basically results from the fact that arithmetic coding provides a mechanism, at least conceptually, to map any given value of a probability estimate in a more or less direct way to a portion of the resulting codeword. Being provided with such an interface, arithmetic coding allows for a clean separation between the tasks of probability modeling and probability estimation, on the one hand, and the actual entropy coding, i.e., the mapping of symbols to codewords, on the other hand.

We present a novel entropy coding concept that combines the advantages of both VLC-based coding and arithmetic coding. To this end, non-binary source symbols are first binarized, i.e., bijectively mapped to strings of binary symbols. Given the binary symbols and associated binary pmfs with values in the unit interval, the key to our proposed design is the partitioning of the unit interval into a small number of disjoint intervals. Each binary symbol is then assigned to one of those so-called *probability intervals*, depending on its pmf. All binary symbols that are assigned to a particular probability interval are coded with a fixed representative probability. This is achieved by the design and application of relatively simple variable-length codes for representing variable-length sequences of binary symbols. Conceptually, this so-called

D. Marpe and H. Schwarz are with the Image Processing Department, Fraunhofer Institute for Telecommunications — Heinrich Hertz Institute, Berlin, Germany, e-mail: {marpe,hschwarz}@hhi.de

T. Wiegand is jointly affiliated with the Image Processing Department, Fraunhofer Institute for Telecommunications — Heinrich Hertz Institute and the Image Communication Chair, Technical University of Berlin, Germany, e-mail: wiegand@hhi.de

probability interval partitioning entropy (PIPE) coding process also permits a decoupling of probability modeling and the actual entropy coding stage in such a way that backward-driven probability estimation and modeling concepts as previously used exclusively in combination with arithmetic coding can be re-used in our proposed PIPE coding framework without any changes.

When preparing this paper, we became aware of the independently developed work of He et al. [13], which is also based on a combination of ideas from both VLC-based and arithmetic coding. However, the main difference between the aforementioned work and our approach is that we are introducing and making use of the probability interval partitioning principle. Without that, [13] is, for instance, restricted to the application of as many and individually tuned VLC tables as given by the number of different probability states which are generated by the probability estimation process.

This paper is organized as follows. The next section provides an overview of PIPE codes. Section III describes the bijective mapping of the coding symbols to binary symbols. The core of the new entropy codec, i.e., the probability interval partitioning and the assignment of the binary symbols to probability intervals is discussed in Section IV. In Section V, simple codes for binary entropy coding at fixed probabilities are described. The overall structure for an encoder and decoder together with multiple multiplexing strategies for the PIPE codewords is described in Section VI.

II. Overview of PIPE coding

We consider the problem of entropy coding of a sequence of coding symbols $\boldsymbol{s} = (s_0, s_1, \ldots, s_{N-1})$. The problem may be compounded by the fact that $\boldsymbol{s}$ can be composed of different sub-sequences $\boldsymbol{s}_l$ with each element of $\boldsymbol{s}_l$ being the outcome of a random variable S_l over an alphabet $\mathcal{A}_l = \{a_0^l, a_1^l, \ldots\}$ with an associated pmf given by the values $P(S_l = a_m^l)$. It is assumed that the probability estimates $P(S_l = a_m^l)$, which may be fixed or variable, are known to encoder and decoder.

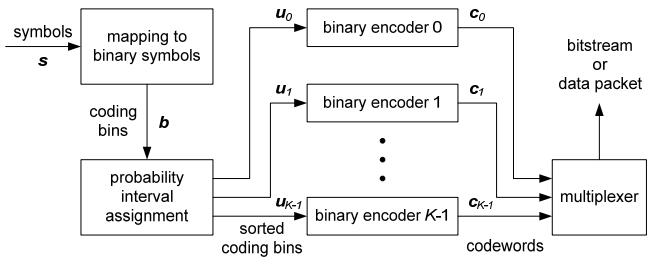

Fig. 1. Overview of the PIPE coding structure.

Figure 1 provides an overview of PIPE coding, which consists of the following steps:

1) The sequence of *coding symbols* $\boldsymbol{s} = (s_0, s_1, \ldots, s_{N-1})$ is bijectively mapped to a sequence of binary symbols $\boldsymbol{b} = (b_0, b_1, \ldots, b_{B-1})$ with the elements of $\boldsymbol{b}$ referred to as *coding bins*. The coding bins b_i are defined in such a way that $P(b_i = 0) \leq 0.5$ holds.[1]
2) The half-open interval $(0, 0.5]$ covering all probability values $P(b_i = 0)$ is partitioned into K disjoint probability intervals $\mathcal{I}_k$ with interval boundaries $(p_k, p_{k+1}]$. Coding bins b_i are assigned to interval $\mathcal{I}_k$ when $p_k < P(b_i = 0) \leq p_{k+1}$, resulting in the sequence of coding bins $\boldsymbol{b} = (b_0, b_1, \ldots, b_{B-1})$ being decomposed into K sub-sequences of coding bins $\boldsymbol{u}_k$, one for each probability interval $\mathcal{I}_k$.
3) The sub-sequence of coding bins $\boldsymbol{u}_k$ collected for an interval $\mathcal{I}_k$ is coded using a fixed representative probability $p_{\mathcal{I}_k}$, thereby allowing the use of simple entropy coding methods such as variable-to-variable length codes to produce a codeword stream $\boldsymbol{c}_k$ for each probability interval $\mathcal{I}_k$.
4) The K codeword streams $\boldsymbol{c}_k$ are multiplexed in order to produce a bitstream or data packet using one of various possible multiplexing scenarios (see Sec. VI).

In the following, a detailed description of each of the individual steps is given. Appendix A provides a worked-through example for PIPE coding.

[1] With a slight abuse of notation, we will use the binary symbol b instead of its related binary-valued random variable B in expressions like $P(B = 0)$.

III. Mapping of Coding Symbols to Coding Bins

The sequence of coding symbols $\boldsymbol{s} = (s_0, s_1, \ldots, s_{N-1})$ is bijectively mapped to a sequence of coding bins $\boldsymbol{b} = (b_0, b_1, \ldots, b_{B-1})$. First, the value of each symbol s_n is mapped to a bin sequence $\boldsymbol{d}_n$ of one or more bins

$$\boldsymbol{d}_n = (d_0^n, \ldots) = \gamma_d^n(s_n) \tag{2}$$

and the resulting bin sequences $\boldsymbol{d}_n$ are concatenated to a bin sequence $\boldsymbol{d} = (d_0, d_1, \ldots, d_{B-1})$. The binarization mapping γ_d^n can be different for different associated random variables S_l. Each bin d_i of the concatenated sequence $\boldsymbol{d}$ is associated with a binary pmf with values in $\{P(d_i = 0), P(d_i = 1)\}$, which can be derived from the pmf of the corresponding source alphabet $\mathcal{A}_l$ and the binarization mapping. For the purpose of coding it is often more convenient to employ the less probable bin (LPB) value $v_{\text{LPB}}^i \in \{0, 1\}$ which is defined as

$$v_{\text{LPB}}^i = \begin{cases} 0, & \text{if } P(d_i = 0) \leq 0.5 \\ 1, & \text{if } P(d_i = 0) > 0.5 \end{cases} \tag{3}$$

together with its corresponding probability estimate

$$p_{\text{LPB}}^i = \begin{cases} P(d_i = 0), & \text{if } P(d_i = 0) \leq 0.5 \\ 1 - P(d_i = 0), & \text{if } P(d_i = 0) > 0.5. \end{cases} \tag{4}$$

The sequence of bins $\boldsymbol{d}$ is converted into a sequence of coding bins $\boldsymbol{b}$ by applying the mapping

$$b_i = d_i \oplus v_{\text{LPB}}^i, \tag{5}$$

for each bin d_i, where $\oplus$ denotes the exclusive or operator. As a consequence, it holds $P(b_i = 0) = p_{\text{LPB}}^i$. At the decoder side, the bins d_i can be uniquely derived by using $d_i = b_i \oplus v_{\text{LPB}}^i$.

This so-called *binarization* process is a usual pre-processing step in many lossless compression algorithms (e.g., [14]) because it simplifies the process of estimating conditional probabilities as well as the subsequent entropy coding stage (in [14] realized by a table-based binary arithmetic coding engine). In any case, this mapping to a binary alphabet is essential for the next step, the core of PIPE coding.

IV. Probability Interval Partitioning and Assignment of Coding Bins

The sequence of coding bins $\boldsymbol{b} = (b_0, b_1, \ldots, b_{B-1})$ does uniquely represent the sequence of source symbols $(s_0, s_1, \ldots, s_{N-1})$, and the corresponding probability estimates, which can be employed for entropy coding, are completely described by the LPB probabilities p^i_{LPB} (with $p^i_{\mathrm{LPB}} \leq 0.5$). Hence, only probabilities in the half-open interval $(0, 0.5]$ need to be considered for designing the binary entropy coder to be applied to the coding bins b_i.

First, for the actual binary entropy coding, the sequence of coding bins $(b_0, b_1, \ldots, b_{B-1})$ is mapped to a small number of probability intervals $\mathcal{I}_k$. For that, the probability interval $(0, 0.5]$ is partitioned into K disjoint intervals $\mathcal{I}_k = (p_k, p_{k+1}]$

$$\bigcup_{k=0}^{K-1} \mathcal{I}_k = (0, 0.5] \quad \text{and} \quad \mathcal{I}_k \cap \mathcal{I}_j = \emptyset \ \text{ for } \ k \neq j. \tag{6}$$

The set of K intervals is characterized by $K-1$ interval borders p_k with $k = 1, \ldots, K-1$. Without loss of generality, we assume $p_k < p_{k+1}$ for $k = 0, \ldots, K-1$. The outer interval borders are fixed and given by $p_0 = 0$ and $p_K = 0.5$.

Then, a simple non-adaptive binary entropy coder is used for each interval $\mathcal{I}_k$. All coding bins b_i with associated probabilities $p^i_{\mathrm{LPB}} \in \mathcal{I}_k$ are assigned to the interval $\mathcal{I}_k$ and are coded with an entropy coder that is optimized for a fixed probability $p_{\mathcal{I}_k}$. In other words, the sequence of coding bins $\boldsymbol{b} = (b_0, \ldots, b_{B-1})$ is decomposed into K separate sequences of coding bins

$$\boldsymbol{u}_k = (u^k_0, u^k_1, \ldots) = (b_i \ : \ b_i \in \boldsymbol{b}, \ p^i_{\mathrm{LPB}} \in \mathcal{I}_k). \tag{7}$$

that are separately coded. Each sequence $\boldsymbol{u}_k$ contains the coding bins in the same order as the original sequence of coding bins $\boldsymbol{b}$, but without those coding bins for which $p^i_{\mathrm{LPB}} \notin \mathcal{I}_k$.

A. Probability Interval Partitioning

For investigating the impact of the probability interval partitioning on the coding efficiency, we assume that an entropy coder for the fixed representative probability $p_{\mathcal{I}_k} \in \mathcal{I}_k$ for an interval $\mathcal{I}_k = (p_k, p_{k+1}]$ achieves the entropy limit for this representative probability. Under these assumptions, the rate R (in bits) for coding a bin with probability p using the optimal entropy coder for the interval representative $p_{\mathcal{I}_k}$ is given by

$$\begin{aligned} R(p, p_{\mathcal{I}_k}) &= -p \ \log_2(p_{\mathcal{I}_k}) - (1-p) \log_2(1 - p_{\mathcal{I}_k}) \\ &= H_b(p_{\mathcal{I}_k}) + (p - p_{\mathcal{I}_k}) \ H'_b(p_{\mathcal{I}_k}), \end{aligned} \tag{8}$$

where $H_b(p)$ represents the binary entropy function

$$H_b(p) = -p \ \log_2 p - (1-p) \ \log_2(1-p) \tag{9}$$

and $H'_b(p)$ its first derivative

$$H'_b(p) = \log_2\left(\frac{1-p}{p}\right). \tag{10}$$

We further assume that the expected values for the relative frequencies of the coding bin probabilities p inside a bin sequence are specified by the probability density function (pdf) $f(p)$, with $\int_0^{0.5} f(p) \ \mathrm{d}p = 1$. For probability models with a countable number of states, the expected values for the relative frequencies could also be specified by a pmf. But since a pmf can be converted into a pdf using the Dirac delta function, we restrict the following investigations on the more general case of a continuous distribution. Then, the average rate $\overline{R}$, in bits per bin, for a given set of K intervals $\{\mathcal{I}_k\}$ with representative probabilities $\{p_{\mathcal{I}_k}\}$ can be written as

$$\overline{R} = E\{R\} = \sum_{k=0}^{K-1} \left(\int_{p_k}^{p_{k+1}} R(p, p_{\mathcal{I}_k}) \ f(p) \ \mathrm{d}p \right). \tag{11}$$

The first partial derivative with respect to any representative probability $p_{\mathcal{I}_k}$, with $k = 0, \ldots, K-1$, is given by

$$\frac{\partial}{\partial p_{\mathcal{I}_k}} \overline{R} = \frac{p_{\mathcal{I}_k} \ \int_{p_k}^{p_{k+1}} f(p) \ \mathrm{d}p - \int_{p_k}^{p_{k+1}} p \ f(p) \ \mathrm{d}p}{p_{\mathcal{I}_k} \ (1 - p_{\mathcal{I}_k}) \ \ln 2}. \tag{12}$$

The equation $\frac{\partial}{\partial p_{\mathcal{I}_k}} \overline{R} = 0$ has a single solution

$$p^*_{\mathcal{I}_k} = \frac{\int_{p_k}^{p_{k+1}} p \ f(p) \ \mathrm{d}p}{\int_{p_k}^{p_{k+1}} f(p) \ \mathrm{d}p} \tag{13}$$

for the representative probability $p_{\mathcal{I}_k}$ inside the domain of definition $\mathcal{I}_k$. The second partial derivative for this solution

$$\frac{\partial^2}{\partial p^2_{\mathcal{I}_k}} \overline{R}(p^*_{\mathcal{I}_k}) = \frac{\int_{p_k}^{p_{k+1}} f(p) \ \mathrm{d}p}{p^*_{\mathcal{I}_k} \ (1 - p^*_{\mathcal{I}_k}) \ \ln 2} \tag{14}$$

is always greater than zero if

$$\int_{p_k}^{p_{k+1}} f(p) \ \mathrm{d}p > 0. \tag{15}$$

Hence, if condition (15) is fulfilled, the value $p^*_{\mathcal{I}_k}$ given in eq. (13) is the representative probability for an interval $\mathcal{I}_k$ that minimizes the average rate $\overline{R}$ given the interval boundaries p_k and p_{k+1}. Otherwise, no bin is projected to the interval $\mathcal{I}_k$ and the representative probability $p_{\mathcal{I}_k} \in \mathcal{I}_k$ can be arbitrarily chosen without any impact on the average rate $\overline{R}$; but such a configuration should be avoided, since the interval $\mathcal{I}_k$ would not be employed for entropy coding.

For finding a condition for optimal interval borders, we investigate the first derivatives of the average rate $\overline{R}$ with respect to the interval borders p_k with $k = 1, \ldots, K-1$. If $f(p) > 0$ for all $p \in [p_{\mathcal{I}_{k-1}}, p_{\mathcal{I}_k})$, the equation $\frac{\partial}{\partial p_k} \overline{R} = 0$ has a single solution

$$p^*_k = \frac{H(p_{\mathcal{I}_k}) - p_{\mathcal{I}_k} H'(p_{\mathcal{I}_k}) - H(p_{\mathcal{I}_{k-1}}) + p_{\mathcal{I}_{k-1}} H'(p_{\mathcal{I}_{k-1}})}{H'(p_{\mathcal{I}_{k-1}}) - H'(p_{\mathcal{I}_k})} \tag{16}$$

for the interval border p_k inside the domain of definition $[p_{\mathcal{I}_{k-1}}, p_{\mathcal{I}_k})$ and the second partial derivative for this solution

$$\frac{\partial^2}{\partial p^2_k} \overline{R}(p^*_k) = f(p^*_k) \left(H'(p_{\mathcal{I}_{k-1}}) - H'(p_{\mathcal{I}_k}) \right) \tag{17}$$

is always greater than zero, so that p^*_k is the interval border $p_k \in [p_{\mathcal{I}_{k-1}}, p_{\mathcal{I}_k})$ that minimizes the average rate $\overline{R}$ given the interval representatives $p_{\mathcal{I}_{k-1}}$ and $p_{\mathcal{I}_k}$. If probabilities $p \in [p_{\mathcal{I}_{k-1}}, p_{\mathcal{I}_k})$ with $f(p) = 0$ exist, the equation $\frac{\partial}{\partial p_k} \overline{R} = 0$ has multiple solutions, but p^*_k given in eq. (16) is still optimal

even though further optimal solutions may exist. It should be noted that the optimal interval border p_k^* represents the intersection point of the two functions $R(p, p_{\mathcal{I}_{k-1}})$ and $R(p, p_{\mathcal{I}_k})$

$$p_k^* = p \quad \text{with} \quad R(p, p_{\mathcal{I}_{k-1}}) = R(p, p_{\mathcal{I}_k}). \tag{18}$$

Given the number of intervals K and the pdf $f(p)$, the interval borders p_k, with $k = 1, \ldots, K-1$, and the interval representatives $p_{\mathcal{I}_k}$, with $k = 0, \ldots, K-1$, that minimize the average rate $\overline{R}$ can be obtained by solving the equation system given by eqs. (13) and (16) subject to the conditions (15) for $k = 0, \ldots, K-1$. This can be achieved with the following iterative algorithm.

Algorithm 1:

1) Partition the interval $(0, 0.5]$ into K arbitrary intervals $\mathcal{I}_k = (p_k, p_{k+1}]$ with $p_0 = 0$, $p_K = 0.5$, and $p_k < p_{k+1}$ for all $k = 0, \ldots, K-1$ in a way that the conditions (15) are obeyed for all $k = 0, \ldots, K-1$.
2) Update the representatives $p_{\mathcal{I}_k}$ with $k = 0, \ldots, K-1$ according to eq. (13).
3) Update the interval borders p_k with $k = 1, \ldots, K-1$ according to eq. (16).
4) Repeat the previous two steps until convergence.

Figure 2 shows an example for a probability interval partitioning that is obtained with the described algorithm. For this example, we assumed a uniform pdf $f(p) = 2$ for $0 < p \leq 0.5$ and partitioned the probability interval $(0, 0.5]$ into $K = 4$ intervals. It can be seen that this partitioning leads to a piecewise linear approximation $R(p, p_{\mathcal{I}_k})|_{\mathcal{I}_k}$ of the binary entropy function $H(p)$ with $R(p, p_{\mathcal{I}_k}) \geq H(p)$ for all $p \in \mathcal{I}_k$.

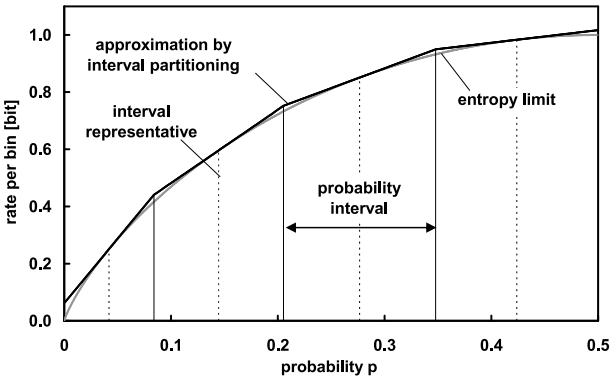

Fig. 2. Example for a probability interval partitioning into $K = 4$ intervals assuming a uniform pdf in $(0, 0.5]$.

B. Excess Rate for Probability Interval Partitioning

As measure for the impact of the interval discretization on the coding efficiency the increase of the average rate relative to the entropy limit

$$\bar{\varrho} = \frac{E\{R\} - E\{H\}}{E\{H\}} = \frac{\overline{R}}{\int_0^{0.5} H(p)\, f(p)\, \mathrm{d}p} - 1 \tag{19}$$

can be used. For the particular example of Figure 2, the expected value of the entropy $E\{H\} = \int_0^{0.5} H(p)\, f(p)\, \mathrm{d}p$ is equal to $1/(2 \ln 2)$ bit per bin and the rate overhead $\bar{\varrho}$ is equal to 1.01%. Table I lists rate overheads $\bar{\varrho}_{\text{uni}}$ and $\bar{\varrho}_{\text{lin}}$ for the uniform pdf and a linear increasing pdf $f(p) = 8p$ with $p \in (0, 0.5]$, respectively, for selected numbers of intervals K.

TABLE I
RATE OVERHEAD VS. THE NUMBER OF PROBABILITY INTERVALS FOR THE UNIFORM AND A LINEAR INCREASING PDF $f(p)$.

K	1	2	4	8	12	16
$\bar{\varrho}_{\text{uni}}$ [%]	12.47	3.67	1.01	0.27	0.12	0.07
$\bar{\varrho}_{\text{lin}}$ [%]	5.68	1.77	0.50	0.14	0.06	0.04

The results given in Table I show that the partitioning of the probability interval $(0, 0.5]$ into a small number of intervals $\mathcal{I}_k$ (e.g., 8 to 16 intervals) with each interval being representated by a fixed probability $p_{\mathcal{I}_k}$ has a very small impact on coding efficiency.

C. Unique Decodability

PIPE coding adds a new mechanism towards unique decodability, which we describe now. It is obvious that any bijective mapping from a sequence of source symbols $\boldsymbol{s}$ to a sequence of coding bins $\boldsymbol{b}$ is invertible. However, in PIPE coding, the sequence of coding bins $\boldsymbol{b}$ is partitioned into K sub-sequences $\boldsymbol{u}_k$, with $k = 0, \ldots, K-1$,

$$\{\boldsymbol{u}_0, \ldots, \boldsymbol{u}_{K-1}\} = \gamma_p(\boldsymbol{b}) \tag{20}$$

and to each of the sub-sequences $\boldsymbol{u}_k$, a sequence of codewords $\boldsymbol{c}_k(\boldsymbol{u}_k)$ is assigned. Consequently, the conditions on unique decodability need to be extended. A sequence of coding bins $\boldsymbol{b}$ is uniquely decodable given K sequences of codewords $\boldsymbol{c}_k(\boldsymbol{u}_k)$, with $k = 0, \ldots, K-1$, if each sub-sequence of coding bins $\boldsymbol{u}_k$ is uniquely decodable given the corresponding codeword sequence $\boldsymbol{c}_k(\boldsymbol{u}_k)$ and the partitioning rule γ_p is known to the decoder. The partitioning rule γ_p is given by the probability interval partitioning $\{\mathcal{I}_k\}$ and the probabilities p^i_{LPB} that are associated with the coding bins b_i, with $i = 0, \ldots, B-1$. Hence, the probability interval partitioning $\{\mathcal{I}_k\}$ has to be known at the decoder side and the probability p^i_{LPB} for each coding bin b_i, with $i = 0, \ldots, B-1$, has to be derived in the same way at encoder and decoder side.

V. Entropy Coding for Probability Intervals

In the following, we first show how a simple but yet efficient code can be designed for fixed probabilities. Given these results, we develop an algorithm that jointly optimizes the code design and the partitioning of the probability interval $(0, 0.5]$.

A. Entropy Coding with Fixed Probabilities

Assuming fixed probabilities $p = p_{\mathcal{I}_k}$ for each probability interval $\mathcal{I}_k$, entropy coding of the corresponding sub-sequences of coding bins $\boldsymbol{u}_k$ can be realized by using binary arithmetic coding or variable length coding. For the latter case, the following approach appears to be simple and efficient.

We consider a binary entropy coding scheme by which a variable number of bins is mapped onto variable length

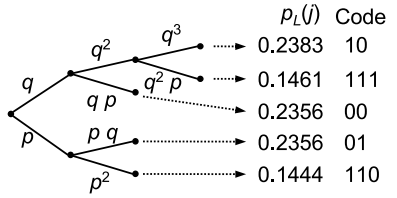

Fig. 3. Tree of binary events for an LPB probability of $p = 0.38$ with an associated variable length code obtained by the Huffman algorithm. The MPB probability $1 - p$ is abbreviated with q.

codewords. For unique decodability, the inverse mapping of a codeword to a bin sequence must be unique. And since we want to design a code that approaches the entropy limit as close as possible, we constrain our considerations to bijective mappings. Such a bijective mapping can be represented by a binary tree where all leaf nodes are associated with codewords, as depicted in Figure 3. The tree branches represent binary events. In the example of Figure 3, the lower branches represent the LPB value and the upper branches represent the more probable bin (MPB) value. The binary tree represents a prefix code for the bins if it is a full binary tree, i.e., if every node is either a leaf or has two descendants. Each node is associated with a probability. The root node has the probability $p_{\text{root}} = 1$. The probability for all other nodes is obtained by multiplying the probability of the corresponding ancestor with p for the LPB descendants and $1 - p$ for the MPB descendants. Each leaf node $\mathcal{L}_j$ is characterized by the number of LPB branches x_j and the number MPB branches y_j from the root node to the leaf node. For a particular LPB probability p, the probability $p_{\mathcal{L}}(j)$ for a leaf node $\mathcal{L}_j = \{x_j, y_j\}$ is equal to

$$p_{\mathcal{L}}(j) = p^{x_j}\ (1-p)^{y_j}. \tag{21}$$

The binary tree $\mathcal{T}$ is characterized by the number of leaf nodes L and the associated pairs $\{x_j, y_j\}$ with $j = 0, \ldots, L-1$.

Given a full binary tree $\mathcal{T}$ and an LPB probability p, the optimal assignment of codewords to the leaf nodes can be obtained by the Huffman algorithm [8]. The resulting variable number of bins to variable length codewords (V2V) mapping $\mathcal{C}$ is characterized by the number of codewords L, which is identical to the number of leaf nodes, and the tuples $\{x_j, y_j, \ell_j\}$, with $j = 0, \ldots, L-1$, where ℓ_j represents the codeword length that is associated with the corresponding leaf node $\mathcal{L}_j = \{x_j, y_j\}$. It should be noted that there are multiple possibilities for the codeword assignment given the codeword lengths ℓ_j. The actual codeword assignment doesn't have any impact as long as the codewords represent a uniquely decodable prefix code. The average rate $R(p, \mathcal{C})$, in bits per bin, for coding a bin with an LPB probability p using a code $\mathcal{C}$ is the ratio of the expected values for the codeword length and the number of bins per codeword

$$\begin{aligned} R(p,\mathcal{C}) &= \frac{\sum_{j=0}^{L-1} p_{\mathcal{L}}(j)\ \ell_j}{\sum_{j=0}^{L-1} p_{\mathcal{L}}(j)\ (x_j + y_j)} \\ &= \frac{\sum_{j=0}^{L-1} p^{x_j}(1-p)^{y_j}\ \ell_j}{\sum_{j=0}^{L-1} p^{x_j}(1-p)^{y_j}(x_j + y_j)}. \end{aligned} \tag{22}$$

The code design is often limited by factors as the maximum number of codewords L, the maximum number of bins per codeword, or the maximum codeword length, or it is restricted to codes of particular structures (e.g., for allowing optimized parsing). If we assume that the set $\mathcal{S}_{\mathcal{C}}$ of usable codes for a particular application is given, the optimum code $\mathcal{C}^* \in \mathcal{S}_{\mathcal{C}}$ for a particular LPB probability p can be found by minimizing the average rate $R(p, \mathcal{C})$

$$\mathcal{C}^*(p) = \arg\min_{\forall \mathcal{C} \in \mathcal{S}_{\mathcal{C}}} R(p, \mathcal{C}). \tag{23}$$

As less complex alternative, the minimization can also proceed over a given set of binary trees $\mathcal{S}_{\mathcal{T}}$ and for each tree only one V2V code $\mathcal{C}$ that is obtained by the Huffman algorithm is considered. As an example, we designed V2V codes for various LPB probabilities p by considering all binary trees $\mathcal{T}$ for which the number of leaf nodes L is less than or equal to a given maximum L_m. In Figure 4, the relative rate increase $\varrho_b(p, \mathcal{C}^*(p)) = R(p, \mathcal{C}^*(p))/H(p) - 1$ is plotted over the LPB probability p for selected maximum table sizes L_m. The rate increase can usually be reduced by allowing larger table sizes. For larger LPB probabilities, a small table size L of 8 to 16 codewords is usually sufficient for keeping the rate increase reasonably small, but for smaller LPB probabilities (e.g., $p < 0.1$), larger table sizes L are required.

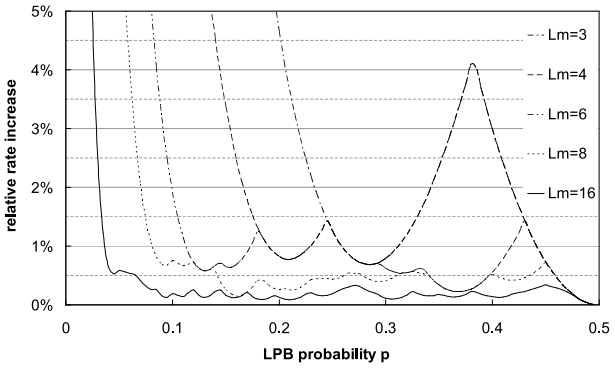

Fig. 4. Relative bit-rate increase $\varrho_b(p, \mathcal{C})$ for optimal codes $\mathcal{C}$ given a maximum number of table entries L_m.

B. Combined Code Design and Interval Partitioning

In the previous sections, we considered the optimal probability interval partitioning assuming optimal codes and the code design for fixed probabilities. But since, in general, we cannot achieve the entropy limit with real V2V codes of limited table sizes, the code design and the partitioning of the probability interval $(0, 0.5]$ must be jointly considered for obtaining an optimized entropy coding design.

For a given interval $\mathcal{I}_k = (p_k, p_{k+1}]$, a code $\mathcal{C}_k$ of a given set $\mathcal{S}_{\mathcal{C}}$ is an optimal code $\mathcal{C}_k^*$ if it minimizes the average rate $\overline{R}|_{\mathcal{I}_k} = \int_{p_k}^{p_{k+1}} R(p, \mathcal{C}_k)\ f(p)\ \mathrm{d}p$ for the given interval $\mathcal{I}_k$.

$$\mathcal{C}_k^* = \arg\min_{\forall \mathcal{C}_k \in \mathcal{S}_{\mathcal{C}}} \int_{p_k}^{p_{k+1}} R(p, \mathcal{C}_k)\ f(p)\ \mathrm{d}p. \tag{24}$$

For practical designs, the minimization of the integral in eq. (24) can be simplified, with only a minor impact on the coding efficiency, by first determining an optimal representative probability $p_{\mathcal{I}_k}^*$ for the interval $\mathcal{I}_k$ according to eq. (13)

and then choosing the optimal code $\mathcal{C}_k^*$ of the given set $\mathcal{S}_\mathcal{C}$ for the representative probability $p_{\mathcal{I}_k}^*$ according to eq. (23).

Optimal interval borders p_k, with $k = 1, \ldots, K-1$, given the set of codes $\mathcal{C}_k$, with $k = 0, \ldots, K-1$, can be derived by minimizing the average rate

$$\overline{R} = E\{R\} = \sum_{k=0}^{K-1} \left(\int_{p_k}^{p_{k+1}} R(p, \mathcal{C}_k)\ f(p)\ \mathrm{d}p \right). \qquad (25)$$

Setting the first derivatives with respect to the interval borders equal to zero, $\frac{\partial}{\partial p_k}\overline{R} = 0$, for $k = 1, \ldots, K-1$, yields

$$p_k^* = p \quad \text{with} \quad R(p, \mathcal{C}_{k-1}) = R(p, \mathcal{C}_k). \qquad (26)$$

Similarly as for eq. (16), it can be shown that p_k^* is always an optimal solution, but depending on the pdf $f(p)$ further optimal solutions might exist. Hence, an optimal interval border p_k^* between two intervals $\mathcal{I}_{k-1}$ and $\mathcal{I}_k$ with given associated codes $\mathcal{C}_{k-1}$ and $\mathcal{C}_k$, respectively, is the intersection point of the functions $R(p, \mathcal{C}_{k-1})$ and $R(p, \mathcal{C}_k)$.

Consequently, the following iterative algorithm can be used for jointly deriving the probability interval partitioning and the associated codes given the number K of probability intervals, the set of usable codes $\mathcal{S}_\mathcal{C}$, and the pdf $f(p)$, with $p \in (0, 0.5]$.

Algorithm 2:

1) Derive initial probability interval boundaries p_k, with $k = 0, \ldots, K$, using Algorithm 1 specified in sec. IV.
2) Derive representatives $p_{\mathcal{I}_k}$ for the probability intervals $\mathcal{I}_k$, with $k = 0, \ldots, K-1$, according to eq. (13).
3) Derive codes $\mathcal{C}_k \in \mathcal{S}_\mathcal{C}$ for the interval representatives $p_{\mathcal{I}_k}$, with $k = 0, \ldots, K-1$, according to eq. (23).
4) Update the interval borders p_k, with $k = 1, \ldots, K-1$, according to eq. (26).
5) Repeat the previous three steps 2–4 until convergence.

The coding efficiency of the derived code set could be improved, when steps 2 and 3 in Algorithm 2 are replaced by a direct derivation of codes $\mathcal{C}_k \in \mathcal{S}_\mathcal{C}$ $(0 \leq k < K)$, based on the interval borders p_k $(0 \leq k \leq K)$, according to eq. (24). And, as mentioned in Sec. V-A, the minimization in step 3 can also proceed over a given set of binary trees $\mathcal{S}_\mathcal{T}$ where for each binary tree $\mathcal{T}$ only one V2V code $\mathcal{C}_k$ obtained by the Huffman algorithm is considered.

As an example, we jointly derived the partitioning into $K = 12$ probability intervals and corresponding V2V codes using Algorithm 2. At this, the minimization in step 3 of the algorithm was replaced with an equivalent minimization over a given set of binary trees $\mathcal{S}_\mathcal{T}$ where the evaluated code $\mathcal{C}$ for each tree $\mathcal{T}$ was obtained by the Huffman algorithm. We considered trees $\mathcal{T}$ with a maximum number of $L_m = 65$ leaf nodes and hence codes $\mathcal{C}$ with up to 65 table entries. All binary trees $\mathcal{T}$ with up to 16 leaf nodes have been evaluated in the minimization; for trees with more than 16 leaf nodes, we employed a suboptimal search given the best results for trees with a smaller number of leaf nodes.

In Figure 5, the rate increase relative to the entropy limit $\Delta R(p) = R(p) - H(p)$ for the code design example is plotted over the LPB probability p. For comparison, we also plotted the rate increase $\Delta R(p)$ for the theoretically optimal probability interval partitioning (as developed in Sec. IV) and the theoretically optimal probability interval partitioning with the additional constraint $p_{\mathcal{I}_{K-1}} = 0.5$. It can be seen that the joint probability interval partitioning and V2V code design leads to a shifting of the interval borders (with the interval borders p_k for $k = 1, \ldots, K-1$ given by the local maxima of the $\Delta R(p)$ curves). The overall rate increase relative to the entropy limit for the design example with real V2V codes is $\bar{\varrho} = 0.24\%$, when assuming a uniform pdf $f(p)$. The corresponding rate increase for the theoretically optimal probability interval partitioning is $\bar{\varrho} = 0.12\%$ (and $\bar{\varrho} = 0.13\%$ with the above mentioned additional constraint).

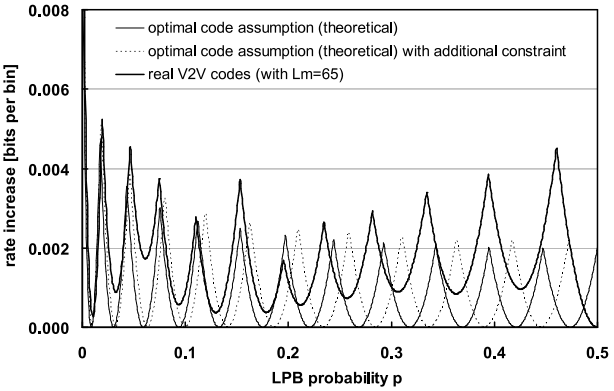

Fig. 5. Rate increase for the theoretically optimal probability interval partitioning into $K = 12$ intervals (cp. Sec. IV) and a real design with V2V codes with a maximum number of $L_m = 65$ table entries. The actual table sizes range between 2 and 65, with an average of 33.

C. Codeword Termination

For a finite sequence of source symbols $(s_0, s_1, \ldots, s_{N-1})$, each of the K binary encoders processes a finite sub-sequence of coding bins $\boldsymbol{u}_k = (u_0^k, u_1^k, \ldots)$, with $k = 0, \ldots, K-1$. It has to be ensured that, for each of the K binary encoders, all coding bins of the sub-sequence $\boldsymbol{u}_k$ can be reconstructed given the corresponding codeword or sequence of codewords $\boldsymbol{c}_k = \boldsymbol{c}_k(\boldsymbol{u}_k)$.

When employing binary arithmetic coding, the arithmetic codeword for each sub-sequence of coding bins has to be terminated in a way that all coding bins can be decoded given the codeword. For the V2V codes described above, the bins at the end of a sub-sequence $\boldsymbol{u}_k$ may not represent a bin sequence that is associated with a codeword. In such a case, any codeword that contains such a terminating bin sequence as prefix can be written. The overhead can be minimized, if the corresponding codeword with minimum length (or one of these codewords) is chosen. At the decoder side, additionally read bins at the end of the bin sequence, which can be identified given the bitstream syntax and the binarization schemes, will be discarded.

VI. PIPE Coding Architecture and Multiplexing

Before discussing approaches for multiplexing the PIPE codewords, we describe the overall PIPE coding architecture.

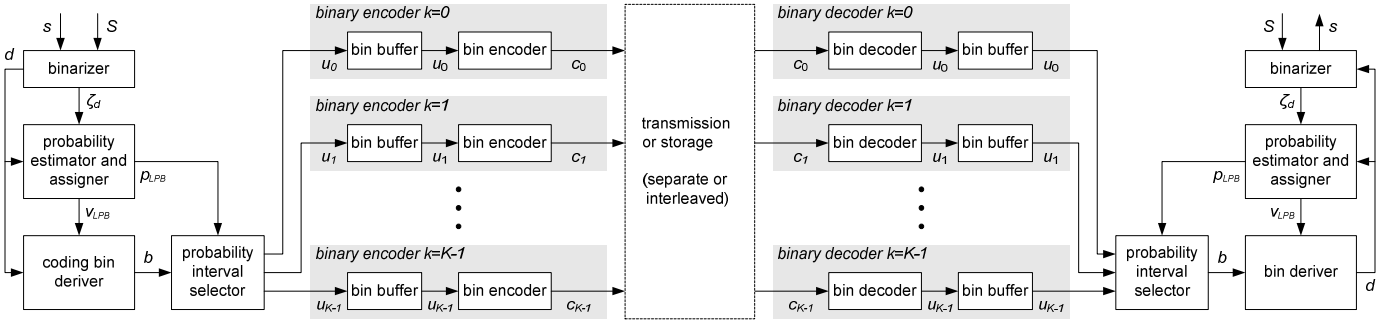

Fig. 6. Block diagram for an example encoder (left part) and decoder (right part).

For that, we concentrate on coding schemes, in which the probabilities p^i_{LPB} for the bins b_i are directly estimated at encoder and decoder side and the K binary coders use V2V mappings described in sec. V-A. We assume that each source symbol s is associated with a random variable S, which determines its underlying alphabet together with its corresponding pmf. The order of symbols and associated random variables shall be given by the syntax, which is presumed to be known at encoder and decoder side.

A. Overview

The block diagram for an example encoder and decoder design is illustrated in Figure 6. At the encoder side, the symbols s together with an indication of the associated random variable S are fed into the *binarizer*, which converts each symbol s into a sequence of bins $\boldsymbol{d} = \gamma_d^S(s)$. The used binarization scheme γ_d^S is determined based on the random variable indication S. In addition, the binarizer associates each bin d of a bin sequence $\boldsymbol{d}$ with a probability model indication ζ_d, which specifies the probability model that is used for coding the bin d. The probability model indication ζ_d can be derived based on the random variable indication S associated with s, the bin number of the current bin inside the bin sequence $\boldsymbol{d}$, and/or the values of already coded bins and symbols.

The *probability estimator and assigner* maintains multiple probability models. It receives bins d and associated probability model indications ζ_d from the binarizer, and forwards the LPB value v_{LPB} and the LPB probability p_{LPB} of the indicated probability model to the coding bin deriver and the probability interval selector, respectively. Thereafter, the corresponding probability model $\{v_{\mathrm{LPB}}, p_{\mathrm{LPB}}\}$ is updated using the value of the received bin d.

The *coding bin deriver* receives bins d and associated LPB values v_{LPB} from the binarizer and the probability estimator and assigner, respectively, and sends coding bins b, which are derived by $b = d \oplus v_{\mathrm{LPB}}$, to the probability interval selector.

The *probability interval selector* forwards each coding bin b to one of the K binary encoders. It contains information about the probability interval partitioning $\{\mathcal{I}_k\}$. The LPB probability p_{LPB}, which is associated with a coding bin b and received from the probability estimator and assigner, is compared to the interval borders $\{p_k\}$ and the probability interval index k, for which $p_{\mathrm{LPB}} \in \mathcal{I}_k$, is derived. Then, the coding bin $u_k = b$ is forwarded to the associated binary encoder.

Each of the K *binary encoders* consists of a *bin buffer* and a *bin encoder*. The bin buffer receives coding bins u_k from the probability interval selector and stores them in coding order. The bin encoder implements a particular V2V mapping and compares the bin sequence in the bin buffer with the bin sequences that are associated with codewords. If the bin sequence in the bin buffer is equal to one of those bin sequences, the bin encoder removes the sequence of bins u_k from the bin buffer and writes the associated codeword c_k to the corresponding codeword stream. At the end of the encoding process for a symbol sequence, for all binary encoders for which the bin buffers are not empty, a terminating codeword is written as described in Sec. V-C.

The K resulting codeword streams can be separately transmitted, packetized, or stored, or they can be interleaved (cp. Sec. VI-B) for the purpose of transmission or storage.

At the decoder side, each of the K binary decoders consisting of a bin decoder and a bin buffer receives one codeword stream. The bin decoder reads codewords c_k from the codeword stream and inserts the associated sequence of coding bins u_k, in coding order, into the bin buffer.

The decoding of the symbol sequence is driven by the underlying syntax. Requests for a symbol s are sent together with the indication of the associated random variable S to the binarizer. The *binarizer* converts these symbol requests into request for bins. A request for a bin is associated with a probability model indication ζ_d, which is derived in the same way as in the encoder, and sent to the probability estimator and assigner. The *probability estimator and assigner* is operated similar to its counterpart at the encoder side. Based on the probability model indication ζ_d, it identifies a probability model and forwards its LPB value v_{LPB} and LPB probability p_{LPB} to the bin deriver and the probability interval selector, respectively.

The *probability interval selector* determines one of the K binary decoders based on the LPB probability p_{LPB}, in the same way as the binary encoder is determined at the encoder side, removes the first coding bin u_k, in coding order, from the corresponding bin buffer, and forwards this coding bin $b = u_k$ to the bin deriver.

The *bin deriver* receives coding bins b and associated LPB values v_{LPB} from the probability interval selector and probability estimator and assigner, respectively, and determines the bin values $d = b \oplus v_{\mathrm{LPB}}$. As final response to a bin request sent by the binarizer, the bin deriver send the decoded bin value d

to the binarizer and the probability estimator and assigner.

In the probability estimator and assigner, the value of the decoded bin d is used to update the probability model $\{v_{\mathrm{LPB}}, p_{\mathrm{LPB}}\}$, which was chosen by the associated value ζ_d, in the same way as at the encoder side. Finally, the binarizer adds the received bin d to the bin sequence $\boldsymbol{d}$ which has been already received for a symbol request and compares this bin sequence $\boldsymbol{d}$ with the bin sequences that are associated with symbol values by the binarization scheme γ_d^S. If the bin sequence $\boldsymbol{d}$ matches one of those bin sequences, the corresponding decoded symbol s is output as final response to the symbol request. Otherwise, the binarizer sends further bin requests until the symbol s is decoded.

The decoding of a symbol sequence is terminated if no further symbol requests, which are driven by the syntax, are received. The coding bins u_k that may be contained in one or more of the K bin buffers at the end of the entropy decoding process (as a result of codeword termination) are discarded.

B. Transmission or Storage of Partial Bitstreams

The K partial bitstreams (or codeword streams) that are generated by the different binary encoders for a set of source symbols (such as a slice of a video picture or a frame of audio samples) can be written to different data packets or different partitions of a data packet. This enables a parallelization of the bin encoding and decoding process. At the encoder side, the coding bins u_k are written to the K bin buffers and the actual bin encoding can be done in parallel. At the decoder side, the K partial bitstreams can be decoded in parallel. The remaining entropy decoding process simply reads the coding bins $b = u_k$ from the corresponding K bin buffers.

The separate transmission or storage of the partial bitstreams requires the signaling of partitioning information or the inclusion of additional data packet headers. If the number of bits for a set of source symbols is small, this side information can form a non-negligible part of the overall bit rate. Furthermore, parallelized entropy encoding and decoding is often not required for small data packets. If the bins are coded using V2V codes and the support of parallelized entropy encoding and decoding is not required, the codewords of the K partial bitstreams can be interleaved without any rate overhead.

At the decoder side, the entire entropy decoding process including the reading of codewords from the bitstream is then controlled by requests for coding symbols. With the exception of the actual bin decoding, the decoding process is operated as described in Sec. VI-A. The probability interval selector determines one of the K binary decoders based on the LPB probability p_{LPB} and sends a bin request to this binary decoder. If the associated bin buffer is empty, the bin decoder reads a new codeword from the bitstream and inserts the corresponding sequence of coding bins into the bin buffer. The probability interval selector removes the first coding bin u_k from the bin buffer and forwards this coding bin $b = u_k$ to the bin deriver as described in Sec. VI-A. The reading of codewords from the bitstream is driven by the bin requests that are sent to the binary decoders.

The encoder must ensure that the order in which the codewords are written to the bitstream is the same as the order in which the codewords are read at the decoder side. This can be realized by introducing a codeword buffer, which represents a first-in first-out buffer with variable-length buffer entries, as illustrated in Figure 7. The encoding process is operated as described in Sec. VI-A with the exception that the binary encoders don't write the codewords to separate partial bitstreams. If a coding bin u_k is send to a particular binary encoder and the associated bin buffer is empty, this binary encoder reserves the next free buffer entry in the codeword buffer before it inserts the coding bin into the bin buffer. When the sequences of coding bins in a bin buffer represents a sequence of coding bins that is associated with a codeword, the codeword is written to the previously reserved buffer entry. Finally, the codewords of the codeword buffer are written to the bitstreams in the order in which the corresponding buffer entries have been reserved by the binary encoders. After a codeword is written to a buffer entry, the codewords of the completed buffer entries at the beginning of the codeword buffer are written to the bitstream until the first buffer entry represents a reserved or free buffer entry. An example for the status of the codeword buffer is illustrated in Figure 7.

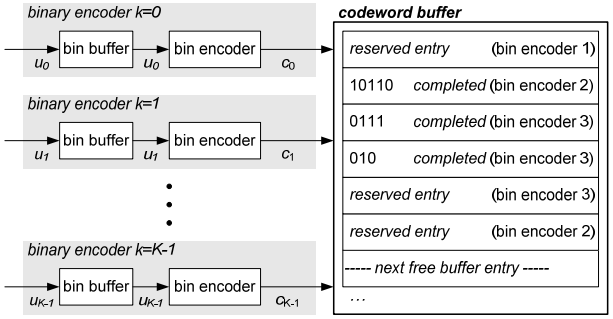

Fig. 7. Interleaving of partial bitstreams using a codeword buffer

C. Low-Delay Operation for Interleaved Bitstreams

The concept of codeword interleaving described above can be extended by a control mechanism, by which the size of the codeword buffer and the associated buffer delay is restricted.

At the encoder side, each reserved and completed entry of the codeword buffer is associated with a maximum codeword length $\ell_{\max}$, which is given by the corresponding V2V mapping. After a new buffer entry is reserved by a binary encoder, the sum $s_{\max}$ of the maximum codeword lengths $\ell_{\max}$ for all reserved and completed buffer entries is compared to a threshold $t_{\max}$, which specifies the maximum buffer delay in bits. If $s_{\max}$ is greater than $t_{\max}$, the codeword of the first buffer entry is terminated as described in sec. V-C and written to the bitstream until $s_{\max}$ is greater than or equal to $t_{\max}$.

The decoder replicates the control mechanism of the encoder. The coding bins that have been added at the encoder side for terminating codewords are discarded.

VII. Conclusion

We have presented a novel approach to entropy coding which is distinguished by a combination of features that were previously considered to be mutually exclusive and each of

which was attributed to either arithmetic coding or variable-length coding. With arithmetic coding the new PIPE coding approach shares its clean interface between probability modeling and coding as well as its near-optimal coding efficiency. As for the commonalities with variable-length coding, PIPE coding is based solely on a small set of VLC tables, where the cardinality of the chosen table set as well as the size of the tables is independent of the underlying source alphabet size. In addition, and unlike for typical VLC coding, PIPE coding permits all sorts of context-based probability modeling without the need of extending or redesigning the underlying VLC table set. Consequently, PIPE codes can be considered as universal codes and as such to be applicable to any source of digital data.

APPENDIX A
WORKED-THROUGH EXAMPLE FOR PIPE CODING

We provide a simple example of a sequence of symbols $\boldsymbol{s}$, where the elements of $\boldsymbol{s}$ are assumed to be outcomes of two random variables S_0 and S_1. The symbol values related to S_0 represent letters of a first alphabet $\mathcal{A}_0 = \{a_0^0, a_1^0, \ldots, a_4^0\}$ of size $M_0 = 5$ and the symbol values related to S_1 represent letters of a second alphabet $\mathcal{A}_1 = \{a_0^1, a_1^1, \ldots, a_5^1\}$ of size $M_1 = 6$. It is further assumed that the assignment of elements of $\boldsymbol{s}$ to one of the two random variables is governed by a sophisticated syntax such that each of both random variables is selected with a probability of 0.5 on average. The probability distribution for both random variables shall be fixed. The alphabets $\mathcal{A}_0$ and $\mathcal{A}_1$ with associated pmf values $\{P(S_0 = a_m^0) : 0 \leq m < M_0\}$ and $\{P(S_1 = a_m^1) : 0 \leq m < M_1\}$, respectively, are given in Table II.

TABLE II
ALPHABETS $\mathcal{A}_0$ AND $\mathcal{A}_1$ WITH ASSOCIATED PROBABILITY MASS FUNCTIONS AND HUFFMAN CODES FOR THE EXAMPLE SOURCE.

$\mathcal{A}_0 = \{a_m^0\}$	a_0^0	a_1^0	a_2^0	a_3^0	a_4^0
$P(S_0 = a_m^0)$	0.68	0.28	0.02	0.019	0.001
$c_{\text{HC}}(a_m^0)$	'0'	'10'	'110'	'1110'	'1111'
$\ell_{\text{HC}}(a_m^0)$	1	2	3	4	4

$\mathcal{A}_1 = \{a_m^1\}$	a_0^1	a_1^1	a_2^1	a_3^1	a_4^1	a_5^1
$P(S_1 = a_m^1)$	0.6	0.34	0.02	0.02	0.0185	0.0015
$c_{\text{HC}}(a_m^1)$	'0'	'10'	'110'	'1110'	'11110'	'11111'
$\ell_{\text{HC}}(a_m^1)$	1	2	3	4	5	5

Table II additionally specifies the assignment of Huffman codes $c_{\text{HC}}(a_m^l)$ with codeword lengths $\ell_{\text{HC}}(a_m^l)$ to letters a_m^l of the alphabets $\mathcal{A}_l$. The entropies of S_0 and S_1, in bits per symbol, are $H(S_0) = 1.1241$ and $H(S_1) = 1.3177$. The average codeword lengths $\bar{\ell}(S_l) = \sum_{m=0}^{M_l-1} P(S_l = a_m^l)\, \ell(a_m^l)$ for the Huffman codes are $\bar{\ell}_{\text{HC}}(S_0) = 1.38$ bit/symbol and $\bar{\ell}_{\text{HC}}(S_1) = 1.52$ bit/symbol, resulting in corresponding redundancies $\bar{\varrho}(S_l) = \bar{\ell}(S_l)/H(S_l) - 1$ of $\bar{\varrho}_{\text{HC}}(S_0) = 22.77\%$ and $\bar{\varrho}_{\text{HC}}(S_1) = 15.36\%$. Given the assumption of uniformly selected symbols from both random variables S_0 and S_1, the overall redundancy of the Huffman coding is $\bar{\varrho}_{\text{HC}} = 18.77\%$.

A. Mapping of Symbols to Coding Bins

For binarizing a symbol sequence $\boldsymbol{s}$, we employ the Huffman codes given in Table II. The bin sequence $\boldsymbol{d}_n$ for a symbol s_n (related to random variable S_l) that represents the alphabet letter a_m^l of the alphabet $\mathcal{A}_l$ is given by the binarization mapping

$$\boldsymbol{d}_n = \gamma_d^n(s_n) = (c_{\text{HC}}(a_m^l) \ : \ s_n \in \mathcal{A}_l,\, s_n = a_m^l). \qquad (27)$$

The particular binarization mappings given in Table II are also referred to as truncated unary binarization schemes [14]. The bin sequences $\boldsymbol{d}_n = (d_0^l, \ldots)$ (with l indicating the associated random variable S_l) consist of one to $M_l - 1$ bins. The bins d_0^l are present for all alphabet letters. A bin d_t^l, with $0 < t < M_l - 1$, is only present if the previous bin d_{t-1}^l is equal to 1. The probability values $P(d_t^l = 0)$, with $0 \leq t < M_l - 1$, for the bins in the binarizations of the letters of an alphabet $\mathcal{A}_l$ can be directly derived from the symbol probabilities $P(S_l = a_m^l)$, with $0 \leq m < M_l$, by $P(d_t^l = 0) = P(S_l = a_t^l) / \sum_{m=t}^{M_l-1} P(S_l = a_m^l)$. These bin probabilities associated with the two symbol alphabets $\mathcal{A}_0$ and $\mathcal{A}_1$ as well as the corresponding probability model descriptions $\{v_{\text{LPB}}^t, p_{\text{LPB}}^t\}$ are summarized in Table III.

TABLE III
BIN PROBABILITIES AND PROBABILITY MODEL DESCRIPTIONS FOR THE BINARIZATION SCHEMES FOR THE ALPHABETS $\mathcal{A}_0$ AND $\mathcal{A}_1$.

d_t^0	d_0^0	d_1^0	d_2^0	d_3^0
$P(d_t^0 = 0)$	0.68	0.875	0.5	0.95
v_{LPB}^t	1	1	0	1
p_{LPB}^t	0.32	0.125	0.5	0.05

d_t^1	d_0^1	d_1^1	d_2^1	d_3^1	d_4^1
$P(d_t^1 = 0)$	0.6	0.85	$0.\overline{33}$	0.5	0.925
v_{LPB}^t	1	1	0	0	1
p_{LPB}^t	0.4	0.15	$0.\overline{33}$	0.5	0.075

As a particular example, we consider the binarization of the symbol sequence $\boldsymbol{s}$ given in Table IV, which consists of two sub-sequences related to S_0 and S_1, respectively, with each containing 5 elements. Table IV lists the symbol values s_n, their associated random variables S_l, and their corresponding bin sequences $\boldsymbol{d}_n$. By a concatenation of these bin sequences $\boldsymbol{d}_n$, the bin sequence $\boldsymbol{d}$, which consists of $B = 20$ bins and uniquely represents the symbol sequence $\boldsymbol{s}$, is obtained.

TABLE IV
EXAMPLE FOR THE BINARIZATION OF A SYMBOL SEQUENCE.

s_n	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9
S_l	S_1	S_0	S_0	S_0	S_1	S_1	S_0	S_1	S_1	S_0
a_m^l	a_1^1	a_0^0	a_0^0	a_3^0	a_0^1	a_0^1	a_0^0	a_5^1	a_0^1	a_2^0
$\boldsymbol{d}_n$	'10'	'0'	'0'	'1110'	'0'	'0'	'0'	'11111'	'0'	'110'

The bin sequence $\boldsymbol{d} = (d_0, d_1, \ldots, d_{B-1})$ is converted into a sequence of coding bins $\boldsymbol{b} = (b_0, b_1, \ldots, b_{B-1})$ by the mapping $b_i = d_i \oplus v_{\text{LPB}}^i$, for $0 \leq i < B$, and each coding bin b_i is associated with a probability $p_{\text{LPB}}^i = P(b_i = 0)$. Table V shows the sequence of bins $\boldsymbol{d}$, the corresponding sequence of coding bins $\boldsymbol{b}$, and the associated probability values p_{LPB}^i for the example of Table IV.

TABLE V
EXAMPLE FOR THE MAPPING OF A SEQUENCE OF BINS TO A SEQUENCE OF CODING BINS WITH ASSOCIATED PROBABILITIES.

index i	0	1	2	3	4	5	6	7	8	9
d_i	1	0	0	0	1	1	1	0	0	0
v^i_{LPB}	1	1	1	1	1	1	0	1	1	1
b_i	0	1	1	1	0	0	1	1	1	1
p^i_{LPB}	0.4	0.15	0.32	0.32	0.32	0.125	0.5	0.05	0.4	0.4
index i	10	11	12	13	14	15	16	17	18	19
d_i	0	1	1	1	1	1	0	1	1	0
v^i_{LPB}	1	1	1	0	0	1	1	1	1	0
b_i	1	0	0	1	1	0	1	0	0	0
p^i_{LPB}	0.32	0.4	0.15	$0.\overline{33}$	0.5	0.075	0.4	0.32	0.125	0.5

B. Probability Interval Partitioning

In the example, only 8 different coding bin probabilities p can occur inside a symbol streams $\boldsymbol{s}$ (cp. Table III). The expected values for the relative frequencies of the bin probabilities can be directly derived given the pmfs for the symbol alphabets as shown in Table II together with the binarization schemes and the assumption that symbols related to each of the two random variables occur with a probability of 0.5. The corresponding pdf $f(p)$ is given by

$$\begin{aligned} f(p) \quad = \quad & 0.0276 \cdot \delta(p - 0.5) + 0.3448 \cdot \delta(p - 0.4) + \\ & 0.0207 \cdot \delta(p - 0.\overline{33}) + 0.3448 \cdot \delta(p - 0.32) + \\ & 0.1379 \cdot \delta(p - 0.15) + 0.1103 \cdot \delta(p - 0.125) + \\ & 0.0069 \cdot \delta(p - 0.075) + 0.0069 \cdot \delta(p - 0.05) \end{aligned} \quad (28)$$

where δ represents the Dirac delta function.

When applying Algorithm 1 with $K = 4$ intervals, the interval boundaries p_k and representative probabilities $p_{\mathcal{I}_k}$ shown in Table VI are obtained. The rate increase for the example source due to the probability interval partitioning is $\bar{\varrho} = 0.12\%$ (assuming optimal entropy codes for the interval representatives $p_{\mathcal{I}_k}$).

TABLE VI
PROBABILITY INTERVAL PARTITIONING FOR THE EXAMPLE SOURCE $\boldsymbol{s}$.

interval $\mathcal{I}_k = (p_k, p_{k+1}]$	$p_{\mathcal{I}_k}$
$\mathcal{I}_0 = (0, 0.0959]$	0.0625
$\mathcal{I}_1 = (0.0959, 0.2206]$	0.1386
$\mathcal{I}_2 = (0.2206, 0.3631]$	0.3208
$\mathcal{I}_3 = (0.3631, 0.5]$	0.4072

The assignment of the coding bins for the example coding bin sequence $\boldsymbol{b} = (b_0, b_1, \ldots, b_{19})$ shown in Table V to the intervals given in Table VI is illustrated in Table VII. With the PIPE approach, the coding bins that are assigned to a particular interval $\mathcal{I}_k$ are coded separately using a code for the representative probability $p_{\mathcal{I}_k}$. For instance, the bin sub-sequence $\boldsymbol{u}_1 = (b_1, b_5, b_{12}, b_{18})$ that is assigned to the interval $\mathcal{I}_1 = (0.0959, 0.2206]$ is coded with a code for the representative probability $p_{\mathcal{I}_1} = 0.1386$.

C. Code Design

In the following, we show examples of simple V2V codes for the representatives $p_{\mathcal{I}_k}$ of the probability intervals $\mathcal{I}_k$ as given in Table VI and being derived for the example source. We have selected V2V codes with a small table size, but a redundancy $\varrho(p_{\mathcal{I}_k})$ of less than 1% for the corresponding interval representative $p_{\mathcal{I}_k}$ $(0 \leq k < 4)$.

TABLE VII
EXAMPLE FOR THE ASSIGNMENT OF CODING BINS TO INTERVALS.

$\mathcal{I}_k$	assigned sub-sequences of coding bins $\boldsymbol{u}_k$
$\mathcal{I}_0$	$\boldsymbol{u}_0 = (b_7, b_{15}) = (1, 0)$
$\mathcal{I}_1$	$\boldsymbol{u}_1 = (b_1, b_5, b_{12}, b_{18}) = (1, 0, 0, 0)$
$\mathcal{I}_2$	$\boldsymbol{u}_2 = (b_2, b_3, b_4, b_{10}, b_{13}, b_{17}) = (1, 1, 0, 1, 1, 0)$
$\mathcal{I}_3$	$\boldsymbol{u}_3 = (b_0, b_6, b_8, b_9, b_{11}, b_{14}, b_{16}, b_{19}) = (0, 1, 1, 1, 0, 1, 1, 0)$

The V2V mapping of bin sequences to codewords for the first representative probability $p_{\mathcal{I}_0}$ is shown in Table VIII, where additionally the actual probability values of the corresponding bin sequences are listed. The redundancy of the chosen V2V code is $\varrho(p_{\mathcal{I}_0}) = 0.89\%$. The V2V codes for the interval representatives $p_{\mathcal{I}_1}$, $p_{\mathcal{I}_2}$, and $p_{\mathcal{I}_3}$ and their corresponding redundancy values are summarized in Table IX.

TABLE VIII
V2V CODE FOR THE REPRESENTATIVE PROBABILITY $p_{\mathcal{I}_0} = 0.0625$ OF THE PROBABILITY INTERVAL PARTITIONING GIVEN IN TABLE VI. THE REDUNDANCY OF THIS CODE IS $\varrho(p_{\mathcal{I}_0}) = 0.89\%$.

bin sequence	probability for bin seq.	codeword
'111111111'	$0.9375^9 = 0.5594$	'1'
'0'	$0.0625^1 = 0.0625$	'0001'
'10'	$0.9375^1 \cdot 0.0625^1 = 0.0586$	'0010'
'110'	$0.9375^2 \cdot 0.0625^1 = 0.0549$	'0010'
'1110'	$0.9375^3 \cdot 0.0625^1 = 0.0515$	'0100'
'11110'	$0.9375^4 \cdot 0.0625^1 = 0.0483$	'0101'
'111110'	$0.9375^5 \cdot 0.0625^1 = 0.0453$	'0110'
'1111110'	$0.9375^6 \cdot 0.0625^1 = 0.0424$	'0111'
'11111110'	$0.9375^7 \cdot 0.0625^1 = 0.0398$	'00000'
'111111110'	$0.9375^8 \cdot 0.0625^1 = 0.0373$	'00001'

TABLE IX
V2V CODES FOR THE REPRESENTATIVE PROBABILITIES $p_{\mathcal{I}_1}$, $p_{\mathcal{I}_2}$, AND $p_{\mathcal{I}_3}$ OF THE PROBABILITY INTERVAL PARTITIONING GIVEN IN TABLE VI.

$p_{\mathcal{I}_1} = 0.1386$, $\varrho(p_{\mathcal{I}_1}) = 0.87\%$	
bin seq.	codeword
'1111'	'1'
'0'	'000'
'10'	'001'
'110'	'010'
'1110'	'011'

$p_{\mathcal{I}_2} = 0.3208$, $\varrho(p_{\mathcal{I}_2}) = 0.55\%$	
bin seq.	codeword
'11'	'1'
'011'	'001'
'101'	'010'
'00'	'011'
'100'	'0000'
'010'	'0001'

$p_{\mathcal{I}_3} = 0.4072$, $\varrho(p_{\mathcal{I}_3}) = 0.71\%$	
bin seq.	codeword
'01'	'01'
'10'	'10'
'111'	'00'
'00'	'111'
'110'	'110'

The overall rate increase relative to the entropy limit by using the PIPE coding approach with 4 intervals as specified in Table VI together with the simple V2V codes of Tables VIII and IX is $\bar{\varrho}_{\text{PIPE}} = 0.80\%$ for the example source. Note that this rate increase is significantly smaller than $\bar{\varrho}_{\text{HC}} = 18.77\%$ as calculated for Huffman coding.

ACKNOWLEDGMENT

The authors would like to thank Bernd Girod, Alexander Keller, Klaus-Robert Müller, and Gary J. Sullivan for com-

ments on early drafts of this paper.

REFERENCES

[1] ITU-T and ISO/IEC JTC1. Digital Compression and Coding of Continuous-Tone Still Images. ISO/IEC 10918-1 — ITU-T Recommendation T.81 (JPEG), September 1992.

[2] ITU-T and ISO/IEC JTC1. JPEG XR image coding system — Part 2: Image coding specification. ISO/IEC 29199-2 — ITU-T Recommendation T.832, 2009.

[3] ISO/IEC. Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s — Part 3: Audio. 1993.

[4] ISO/IEC. Generic coding of moving pictures and associated audio information — Part 7: Advanced Audio Coding (AAC). 2006.

[5] ITU-T Recommendation H.262 — ISO/IEC JTC1 IS 13818-2. Generic Coding of Moving Pictures and Associated Audio Information — Part 2: Video. November 1994.

[6] ITU-T and ISO/IEC. Advanced Video Coding for Generic Audiovisual Services. ITU-T Rec. H.264 and ISO/IEC 14496-10 (MPEG-4 AVC), Version 1: May 2003, Version 2: May 2004, Version 3: Mar. 2005, Version 4: Sep. 2005, Version 5 and Version 6: June 2006, Version 7: Apr. 2007, Version 8 (including SVC): July 2007.

[7] C. E. Shannon. A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27(3):2163–2177, July 1948.

[8] D. A. Huffman. A Method for the Construction of Minimum Redundancy Codes. *Proc. IRE*, pages 1098–1101, September 1952.

[9] N. Abramson. *Information Theory and Coding*. McGraw-Hill, New York, USA, 1963.

[10] J. Rissanen. Generalized Kraft Inequality and Arithmetic Coding. *IBM J. Res. Develop.*, 20:198–203, 1976.

[11] R. Pasco. Source Coding Algorithms for Fast Data Compression. Ph.D. dissertation, Stanford University, 1976.

[12] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley and Sons, Hoboken ,NJ, USA, 2006. 2nd edition.

[13] D. He, G. Korodi, G. Martin-Cocher, E.-h. Yang, X. Yu, and J. Zan. Video coding technology proposal by RIM, Doc. JCTVC-A120. 2010.

[14] D. Marpe, H. Schwarz, and T. Wiegand. Context-Adaptive Binary Arithmetic Coding for H.264/AVC. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):620–636, July 2003.