A new class of codes for robust compression of heterogeneous data:Multiplexed Codes

Hervé Jégou, Christine Guillemot

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

A new class of codes for robust compression of heterogeneous data: Multiplexed Codes

Hervé Jégou — Christine Guillemot

N° 4922

August 2003

THÈME 3

ISSN 0249-6399 ISRN INRIA/RR--4922--FR+ENG

INRIA
RENNES

A new class of codes for robust compression of heterogeneous data: Multiplexed Codes

Hervé Jégou , Christine Guillemot *

Thème 3 — Interaction homme-machine,
images, données, connaissances
Projet TEMICS

Rapport de recherche n° 4922 — August 2003 — 27 pages

Abstract: Compression systems of real signals (images, video, audio) generate sources of information with different levels of priority which are then encoded with variable length codes (VLC). This paper addresses the issue of robust transmission of such VLC encoded heterogeneous sources over error-prone channels. VLCs are very sensitive to channel noise: when some bits are altered, synchronization losses can occur at the receiver. This paper describes a new family of codes, called multiplexed codes, that allow to confine the desynchronization phenomenon to low priority data while allowing to reach asymptotically the entropy bound for both (low and high priority) sources. The idea consists in creating fixed length codes for high priority information and in using the inherent redundancy to describe low priority data, hence the name multiplexed codes. Theoritical and simulation results reveal a very high error resilience at almost no cost in compression efficiency.

Key-words: source coding, variable length codes, data compression, data communication, entropy codes

* Hervé Jégou is with IRISA/ENS Cachan, Christine Guillemot is with IRISA/INRIA

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 Rennes Cedex (France)
Téléphone : +33 2 99 84 71 00 — Télécopie : +33 2 99 84 71 71

Une nouvelle classe de codes pour la compression robuste de données hétérogènes: les Codes Multiplexés

Résumé : Les systèmes de compression de signaux réels (images, vidéo, son) produisent des sources d'informations de priorités différentes, qui sont encodées en utilisant des codes à longueur variable (VLC). Cet article traite du problème de la transmission de telles sources sur des canaux sujets à erreurs. Les codes à longueur variable sont très sensibles au bruit de transmission: lorsque des inversions de bits se produisent, la synchronisation est perdue au décodeur. Une nouvelle famille de codes, les codes multiplexés, permettent de confiner le phénomène de désynchronisation à des données moins prioritaires, tout en permettant d'atteindre la borne entropique pour l'ensemble des données. Le principe consiste à créer un code à longueur fixe pour la source de données prioritaires et d'utiliser la redondance intrinsèque de ce code pour décrire les données moins prioritaires. Les résultats de simulations et théoriques démontrent une résistance aux erreurs très importante pour une perte d'efficacité en compression très faible.

Mots-clés : codage de source, codes à longueur variable, compression de données, communications numériques, codage entropique

1 Introduction

Entropy coding, producing variable length codewords (VLC), is a core component of any data compression scheme. Unlike fixed length codes (FLC), variable length codes are designed to exploit the inherent redundancy of a symbol distribution. The main drawback of VLCs is their high sensitivity to channel noise: when some bits are altered by the channel, synchronization losses can occur at the receiver, the position of symbol boundaries are not properly estimated, leading to dramatic symbol error rates.

This phenomenon has first led some authors to abandon VLCs, and prefer a pre-processing (e.g. non-uniform quantization) providing uniformly distributed symbols, and allowing the usage of fixed length codes. However, the high computational cost of signal-adaptive non-uniform quantizers has in parallel motivated studies of the synchronization ability of VLCs as well as the design of codes with better synchronization properties. The authors in [7] have developed a state model for synchronization recovery suitable for analyzing the performance of various codes with respect to error recovery. It is shown that Huffman codes generated for a given source may have different "re-synchronization" capabilities. In particular, in [13], the authors show that some Huffman codes, tailored to a given source, may contain a codeword that can serve as a "re-synchronization" point. However, the existence of this codeword depends on the source statistics, hence it may not always exist. Modified VLCs, self-synchronizing Huffman codes [8], and reversible VLCs [11, 14, 3], have been designed to fight against desynchronizations. They all add redundancy in the generated bitstream. In video coding standards (H.263, MPEG1-4), synchronization codewords (synchronization markers) are inserted at fixed intervals in the bitstream.

Soft VLC decoding ideas, exploiting residual source redundancy (the so-called "excess-rate"), have also been shown to reduce the "de-synchronization" effect as well as the residual symbol error rates. These ideas rely on capitalizing on source coder suboptimality, by exploiting inner codeword redundancy and exploiting correlation within the sequence of symbols (inter symbol dependency). Models incorporating both VLC-encoded sources and channel codes (CC) have also been considered [9, 5, 3]. The authors in [9] derive a global stochastic automaton model of the transmitted bitstream by computing the product of the separate models (Markov source (MS), source coder (SC) and channel coder (CC)). The authors in [6] push further the above idea by designing an iterative estimation technique alternating the use of the three models (MS, SC, and CC).

The above methods often consist in re-augmenting the redundancy of the bitstream, by introducing an error correcting code or dedicated patterns in the chain. Also, the decoding algorithms, often relying on MAP estimators, have a rather high complexity. Here, we consider instead the design of a new family of codes, called "multiplexed codes", that allow to control (even avoid) the "de-synchronization" phenomenon of VLC encoded sources, while still allowing to reach asymptotically the entropy bound and to rely on simple instantaneous decoding techniques. The principle underlying these codes builds upon the idea that compression systems of real signals generate sources of information with different levels of priority (e.g. texture and motion information for a video signal). This idea underlies unequal error protection (UEP) techniques, which allocate different levels of protection to the

different types of information, either to signal frequency bands [12], to bit planes [10], or to quantization layers [4].

In the wake of this idea, we consider two sources, a high priority source and a low priority source referred respectively as $\mathbf{S_H}$ and $\mathbf{S_L}$ in the sequel. The codes designed are such that the risk of "de-synchronization" is confined to the low priority information. The idea consists in creating a fixed length code (FLCs) for the $\mathbf{S_H}$ source, and in exploiting the inherent redundancy to represent or store information of the low priority source $\mathbf{S_L}$. Hence, the source $\mathbf{S_H}$ inherits some of the properties of FLCs such as random access in the data stream and high error resilience. It is shown that these codes allow to almost reach the entropy bound.

The rest of the paper is organized as follows. Section 2 introduces the notations we use in the sequel. Section 3 outlines the principle of multiplexed codes, discusses their compression properties and describes the first coding algorithm in the case of two sources. The approach relies on a mapping of the low priority flow of data into a sequence of N_i-valued variables, where N_i depends on the realization of the high priority sequence of symbols. A mapping that would be optimal with respect to reaching the entropy bound may require the variable N_i to take its value in a very large set. An optimization aiming at reducing the complexity of the mapping is proposed. It relies on a hierarchical processing of the variables to be computed. Still to reduce the computational cost, a second method is described in section 4. The approach relies on a decomposition on a constrained set of N_i-valued variables (i.e. on a constrained partition of the fixed length codewords into *equivalence classes*) expressed in terms of prime-valued variables. The resulting loss in compression depends on the distance between the respective probability density functions (pdf) of the constrained set of N_i-valued variable and of the source $\mathbf{S_H}$. A heuristic method for selecting the set of N_i-valued variables (i.e., the partition) that would best approximate the pdf of the source $\mathbf{S_H}$ is described in section 5. The impact of channel noise, considering a binary symmetric channel, on the overall source distortion is analyzed in section 6. Last, the choice of the parameters of the algorithm are discussed in section 7 and simulation results are provided in section 8.

2 Problem statement and Notations

Let $\mathbf{S}_H = (S_1, S_2, \ldots, S_t, \ldots, S_{K_H})$ be a sequence of source symbols of high priority taking their values in a finite alphabet $\mathcal{A}$ composed of Ω symbols, $\mathcal{A} = \{a_1, a_2, \ldots, a_i, \ldots, a_\Omega\}$. Note that, in the following, we reserve capital letters to random variables, and small letters to values of these variables. Bold face characters will be used to denote vectors or sequences. The stationary probability of the source $\mathbf{S}_H$ is denoted $\boldsymbol{\mu} = (\mu_1, \mu_2, \ldots, \mu_i, \ldots, \mu_\Omega)$, where μ_i stands for the probability that a symbol of $\mathbf{S_H}$ equals a_i. Let h be the stationary entropy of the source per symbol, given by $h = -\sum_{i=1}^{\Omega} \mu_i \, log_2(\mu_i)$. Let $\mathbf{S}_L = (S'_1, S'_2, \ldots, S'_{i'}, \ldots, S'_{K_L})$ be a sequence of source symbols of lower priority taking their values in a finite alphabet $\mathcal{A}'$. We assume that the realization $\mathbf{s_L}$ of this source has been pre-encoded into a bitstream $\mathbf{b} = (b_1, b_2, \ldots, b_r, \ldots, b_{K_B})$ with a VLC coder (e.g. Huffman or arithmetic coder). The problem addressed here is the design of a family of joint codes for the two sources $\mathbf{S_H}$ and

class $\mathcal{C}_i$	codeword $c_{i,q}$	symbol a_i	N_i=card($\mathcal{C}_i$)	probability μ_i	index q
$\mathcal{C}_1$	0000	a_1	6	0.43	0
	0001				1
	0010				2
	0011				3
	0100				4
	0101				5
$\mathcal{C}_2$	0110	a_2	5	0.30	0
	0111				1
	1000				2
	1001				3
	1010				4
$\mathcal{C}_3$	1011	a_3	4	0.25	0
	1100				1
	1101				2
	1110				3
$\mathcal{C}_4$	1111	a_4	1	0.02	0

Table 1: An example of multiplexed codes ($c = 4$).

$\mathbf{S_L}$ that would guarantee no "de-synchronization" of the high priority source $\mathbf{S_H}$ at almost no cost in terms of compression efficiency.

3 Multiplexed Codes

3.1 Principle

Let c be a fixed number of bits reserved for the representation of any symbol of the alphabet $\mathcal{A}$. The c bits define a set of $N = 2^c$ codewords. This set of codewords is partitioned into Ω subsets, $\mathcal{C}_i, i = 1 \ldots \Omega$, called *equivalence classes* associated to symbols a_i of the alphabet $\mathcal{A}$, as shown in Table 1. The condition $c \geq log_2(\Omega)$ is required to have at least 1 codeword per subset. Each *equivalence class* $\mathcal{C}_i$ contains a set of codewords $\{c_{i,0}, c_{i,1}, \ldots, c_{i,q}, \ldots, c_{i,N_i-1}\}$, where the integer N_i stands for the number of codewords in the subset (cardinality of the *equivalence class*) and is such that $\sum_{i=1}^{\Omega} N_i = N$.

A symbol $S_t = a_i$ of the flow $\mathbf{S}_H$ can be encoded with any c-bit codeword $c_{i,q}$ belonging to the *equivalence class* $\mathcal{C}_i$ (see example of Table 1). Hence, each codeword S_t can be mapped into a pair $(\mathcal{C}_i, Q)$ of two variables denoting respectively the *equivalence class* and the index of the codeword in the *equivalence class* $\mathcal{C}_i$. The variable Q is an N_i-valued variable, taking its value between 0 and $N_i - 1$ (see Table 1) and representing the inherent redundancy of the c-bits fixed length codes. This redundancy will be exploited to describe the lower priority flow of data. Therefore, to the realization of the sequence of symbols $\mathbf{S}_H$ one can associate a sequence of N_i-valued variables $\mathbf{N} = N_1, \ldots N_i, \ldots N_{K_H}$, which will be used to describe

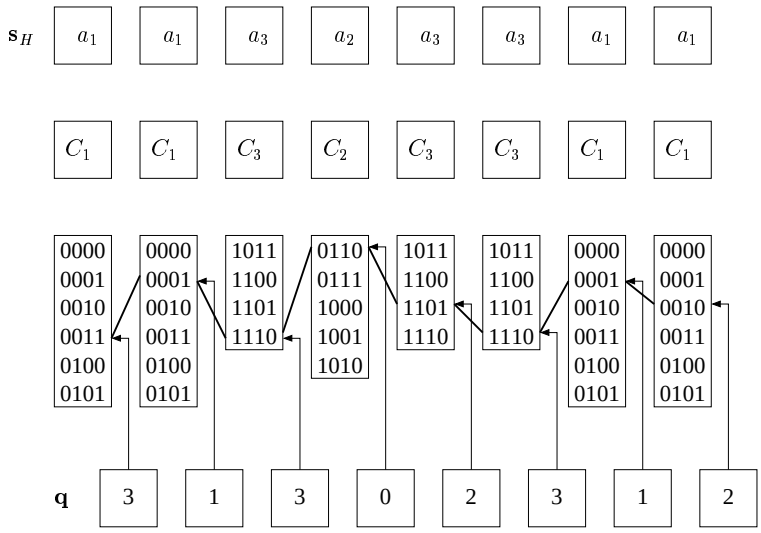


Figure 1: Link between the source $\mathbf{S}_H$, the low priority source formatted as a state flow $\mathbf{q}$ of N_i-valued variables, and multiplexed codewords. Here, the path in red denotes the chosen codewords.

jointly the $\mathbf{S}_H$ and $\mathbf{S}_L$ data flows.

Definition 1: A multiplexed code is defined as the function which maps a c-bits fixed length codeword $c_{i,q}$ into a pair of variables comprising the symbol value a_i of the alphabet $\mathcal{A}$ (on which the high priority source is quantized) and the value q of a variable Q denoting the index of the codeword in the *equivalence class* associated to a_i as:

$$c_{i,q} \leftrightarrows (a_i, q) \tag{1}$$

For example, let $\mathcal{A} = \{a_1, a_2, a_3, a_4\}$ be the alphabet of the source $\mathbf{S}_H$ with the stationary probabilities given by $\mu_1 = 0.43$, $\mu_2 = 0.30$, $\mu_3 = 0.25$, and $\mu_4 = 0.02$. Table 1 gives an example of partitioning of the set of $N = 2^c$ codewords into the 4 *equivalence classes* associated to the alphabet symbols. This partitioning reveals 4 N_i-valued variables, where N_i is the cardinality of each class. Note that, in Table 1, the codes are ranked in the lexicographic order, but the method is not restricted to this order. Knowing the sequence $\mathbf{s}_H$ to be transmitted, it appears that several choices of codewords are available, as depicted in figure 1. Thus, additional variables realizations can be described by the indexes of codewords within the equivalence classes related to the realization $\mathbf{s}_H$. These indexes are referred to as state values $\mathbf{q}$,

3.2 Conversion of the lower priority bitstream

It appears from above that the design of multiplexed codes relies on the choice of the partition $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_\Omega\}$. The encoding then proceeds with the conversion of the lower priority bitstream into a flow of variables taking their values in the different sets of N_i-valued variables.

In order to be multiplexed with symbols of $\mathbf{S}_H$, the lower priority bitstream $\mathbf{b}$ must be mapped into a sequence of N_i-valued variables. Let us consider the mapping of the realization $\mathbf{s}_H = s_1 \ldots s_t \ldots s_{K_H}$ into the sequence $\mathbf{n} = n_1 \ldots n_t \ldots n_{K_H}$. The sequence of n_t-valued variables can also be seen as a unique Λ-valued variable θ, where $\Lambda = \prod_{t=1}^{K_H} n_t$. The variable θ hence verifies $0 \leq \theta \leq \Lambda - 1$. The quantity Λ denotes the number of different multiplexed sequences of codewords $c_{i,q}$ that can be used as a coded representation of the sequence $\mathbf{s}_H$. One of these sequences can be used as a multiplexed coded description of the two sequences $\mathbf{s}_H$ and $\mathbf{s}_L$ [1]. The sequence of multiplexed codewords to be transmitted will then depend on the bitstream representation $\mathbf{b}$ of the source $\mathbf{S}_L$.

The maximum amount of information that can be stored in the Λ-valued variable θ is theoretically $log_2(\Lambda)$ bits. However, since this variable is used to store bits of the bitstream $\mathbf{b}$, only $K'_B = \lfloor log_2(\Lambda) \rfloor$ bits can be stored, leading to an overhead equal to $\frac{log_2(\Lambda)}{K'_B}$.

The last[2] K'_B bits of $\mathbf{b}$ are then seen as the binary representation of an integer γ comprised between 0 and $2^{K'_B} - 1$, that can be expressed as

$$\gamma = \sum_{r=1}^{K'_B} b_{(r+K_B-K'_B)}\ 2^{r-1} \tag{2}$$

The variable γ must then be expanded into a sequence of pairs (n_t, q_t) that will provide entries in the multiplexed codes table. Let us recall that q_t denotes the index of a codeword in the *equivalence class* associated to the realization s_t of a given symbol S_t. There are different ways to expand the variable γ into the sequence of n_t-valued variables (q_t), where $t = 1, \ldots K_H$. One possible method is to use the recursive Euclidean decomposition:

Definition 2: The recursive Euclidean decomposition of a positive integer x by a sequence $\mathbf{y} = (y_1, y_2, \ldots, y_K)$ of positive integers is the unique sequence $\mathbf{r} = (r_1, r_2, \ldots, r_K) \in \mathbb{N}^K$ such that:

$$\begin{cases} x = r_1 + y_1(r_2 + y_2(\ldots(r_t + y_t(\ldots(r_{K-1} + y_{K-1}\, r_K)\ldots))\ldots)) \\ \forall t \in [1..K], 0 \leq r_t \leq y_t - 1 \end{cases} \tag{3}$$

It leads to expand γ as $q_1 + n_1(q_2 + n_2(\ldots(q_{K_H-1} + n_{K_H-1}\, q_{K_H})\ldots))$. The components q_t can then be calculated by the algorithm 1.

In summary, the encoding proceeds as follows:

[1] In fact, only a part of $\mathbf{s_L}$, whose length depends on the multiplexing capacity of the sequence $\mathbf{s_H}$

[2] Any other subset of K'_B bits of $\mathbf{b}$ can be used

Algorithm 1 Conversion of the integer γ into the sequence $\mathbf{q}$

for $t = 1$ to K_H **do**
  $q_t = \gamma'$ *modulo* n_t
  $\gamma' = \frac{\gamma' - q_t}{n_t}$
end for

t	s_t	n_t	γ'	q_t	c_{i,q_t}
1	a_1	6	159 957	3	0011
2	a_1	6	26 659	1	0001
3	a_3	4	4 443	3	1110
4	a_2	5	1 110	0	0110
5	a_3	4	222	2	1101
6	a_3	4	55	3	1110
7	a_1	6	13	1	0001
8	a_1	6	2	2	0010

Table 2: Euclidean decomposition of the variable γ and corresponding multiplexed codewords.

1. For $t = 1, \ldots K_H$, let n_t be the *cardinality* of the *equivalence class* associated to the realization s_t of the symbol S_t.

2. The integer Λ is computed as $\Lambda = \prod_{t=1}^{K_H} n_t$. The number K'_B of bits of the lower priority bitstream $\mathbf{b}$ that can be jointly encoded with the sequence $\mathbf{S}_H$ is given by $\lfloor log_2(\Lambda) \rfloor$.

3. The K'_B last bits of the low priority bitstream $\mathbf{b}$ are converted into the sequence of pairs (n_t, q_t), $t = 1 \ldots K_H$, using the algorithm (1) of the Euclidean decomposition of Eqn. (3).

4. The sequence of pairs (s_t, q_t) provides the entries in the table of multiplexed codes, hence allows to select the sequence of c-bits fixed length codes to be transmitted on the channel.

5. If $K'_B \leq K_B$, the $K_B - K'_B$ first bits of the bitstream $\mathbf{b}$ are then concatenated to the sequence of multiplexed codewords. Otherwise, the remaining symbols of $\mathbf{S_H}$ are not multiplexed but sent using other codes on the channel, such as FLCs or Huffman codes.

As all steps are reversible, the decoding proceeds in the reverse order.

Algorithm 2 Conversion of the integer γ into the sequence of states $\mathbf{q}$ using the hierarchical algorithm (for sequences such that $\exists\, l / K_H = 2^l$).

for $j = 1$ to l **do** {Processing of values $n_{t_1}^{t_2}$ involved in further processing}
  for $i = 1$ to 2^{l-j} **do**
    $t_1 = (i-1)\,2^j + 1$
    $t_2 = i\,2^j$
    $t_m = (t_1 + t_2 - 1)/2$
    $n_{t_1}^{t_2} = n_{t_1}^{t_m}\, n_{t_m+1}^{t_2}$
  end for
end for

for $j = l$ to 1 by -1 **do** {Processing of values q_{t1}^{t2}}
  for $i = 1$ to 2^{l-j} **do**
    $t_1 = (i-1)\,2^j + 1$
    $t_2 = i\,2^j$
    $t_m = (t_1 + t_2 - 1)/2$
    $q_{t_1}^{t_m} = q_{t_1}^{t_2}\ modulo\ n_{t_1}^{t_m}$
    $q_{t_m+1}^{t_2} = (q_{t_1}^{t_2} - q_{t_1}^{t_m})/n_{t_1}^{t_m}$
  end for
end for

3.3 Example

Let us consider again the source alphabet described in section 3 and the multiplexed code given in Table 1. Considering the sequence of symbols $\mathbf{S}_H$ given in Table 2, the algorithm proceeds first with the derivation of the sequence of n_t variables as shown in Table 2. This leads to $\Lambda \approx 2^{18.66}$. Therefore, the last 18 bits of the bitstream $\mathbf{b} = 1010\ 1011\ 0000\ 1110\ 01$ are used to process the variable $\gamma = \sum_{r=1}^{18} b_r.2^{r-1} = 159\ 957$. The Euclidean decomposition of the variable γ according to Eqn. 1 leads to the sequence of q_t variables (indexes of the codewords in the classes of equivalence associated to the sequence of symbols) given in Table 2. The sequence of pairs (s_t, q_t) provides directly the entries in the table of c-bits fixed length codewords, i.e. in the table of multiplexed codes.

3.4 A hierarchical algorithm for the computation of γ

The decomposition of γ involves to deal with long integers. If the long integer γ is computed according to the algorithm 1, i.e. as in the example of Table 2, the complexity can be evaluated as follows. Each step of the algorithm presented in Table 2 computes a modulo operation and a division whose operands are respectively γ' and the integer n_t. Such operations can be processed with $O(K_H)$ elementary operations. It leads to a maximal complexity in the order of $O(K_H^2)$.

Algorithm 3 Conversion of the sequence of states $\mathbf{q}$ into the global state γ using the hierarchical algorithm (for sequences such that $\exists l / K_H = 2^l$).

for $j = 1$ to l **do**
  for $i = 1$ to 2^{l-j} **do**
    $t_1 = (i-1)\,2^j + 1$
    $t_2 = i\,2^j$
    $t_m = (t_1 + t_2 - 1)/2$
    $q_{t_1}^{t_2} = q_{t_1}^{t_m} + n_{t_1}^{t_m}\, q_{t_m+1}^{t_2}$
    $n_{t_1}^{t_2} = n_{t_1}^{t_m}\, n_{t_m+1}^{t_2}$
  end for
end for
$\gamma = n_1^{2^l}$

A hierarchical algorithm allows to reduce this complexity using another decomposition of γ. For sake of clarity, we assume that $K_H = 2^l$. The approach can be easily extended to any length. Let $n_{t_1}^{t_2}$ denotes the product $\prod_{t=t_1}^{t_2} n_t$. Similarly, $q_{t_1}^{t_2}$ is defined as

$$q_{t_1}^{t_2} = q_{t_1} + n_{t_1}(q_{t_1+1} + \ldots(q_{t_2-1} + n_{t_2-1}q_{t_2})\ldots). \tag{4}$$

If the number of terms in Eqn. 4 is a power of two, i.e if $t_2 - t_1 - 1 = 2^{l'}$, then the entities $n_{t_1}^{t_2}$ and $q_{t_1}^{t_2}$ can be respectively decomposed as

$$n_{t_1}^{t_2} = n_{t_1}^{\frac{t_2+t_1-1}{2}}\, n_{\frac{t_2+t_1+1}{2}}^{t_2}, \tag{5}$$

$$q_{t_1}^{t_2} = q_{t_1}^{\frac{t_2+t_1-1}{2}} + n_{t_1}^{\frac{t_2+t_1-1}{2}}\, q_{\frac{t_2+t_1+1}{2}}^{t_2}. \tag{6}$$

Notice that $\gamma = q_1^{2^l}$, hence the state γ can be decomposed as

$$\gamma = q_1^{2^{l-1}} + n_1^{2^{l-1}}\, q_{2^{l-1}+1}^{2^l}. \tag{7}$$

Similarly, each term of Eqn. 7 can be recursively decomposed using Eqn. 5 and Eqn. 6. Thus, the conversion of the integer γ into a sequence of states $\mathbf{q}$ can be done iteratively, as described in algorithm 2. Note that each level j generates the variables required for level $j+1$. The reverse algorithm is also described (algorithm 3). The complexity of this algorithm strongly depends on the complexity of the algorithms used for the multiplication, division and modulo operations. Karatsuba [2] has explicitly shown that these operations have a subquadractic complexity. For example, the multiplication can be done in $O(N^{\frac{log3}{log2}})$ with the Karatsuba algorithm[3]. For implementation purpose, efficient algorithms are described in

[3]Some algorithms, such as the Toom-3 and the FFT multiplication algorithms, perform better.

the GNU Multiple precision computing library [1]. Now, let us assume that the complexity of long integer operations are given by $C(K)$, where K is the block size of the largest variable involved. Then, processing the level j in algorithm 2 has the complexity $2^{l-j}\,C(2^j) = K_H\,2^{-j}C(2^j)$. Assuming that $C(K) = O(K^r)$ with $r > 1$, the overall complexity can be written as

$$K_H \sum_{j=1}^{l} 2^{-j} O((2^j)^r) \tag{8}$$

Or, from $C(K) = O(K^r)$, we deduce that $\exists A > 0\ \exists B > 0\ /\ \forall K \in \mathbb{N} - \{0\},\ C(K) < A + B\,K^r$. This leads to the following inegality

$$K_H \sum_{j=1}^{l} 2^{-j} C(2^j) < K_H \sum_{j=1}^{l} 2^{-j}(A + B\,(2^j)^r) \tag{9}$$

$$< K_H A \sum_{j=1}^{l} 2^{-j} + K_H\,B \sum_{j=1}^{l} 2^{j(r-1)} \tag{10}$$

$$< K_H\,A + K_H\,B\,2^{(l+1)(r-1)} \tag{11}$$

Since we have

$$2^{(l+1)(r-1)} = (K_H)^{r-1}\,2^{r-1}, \tag{12}$$

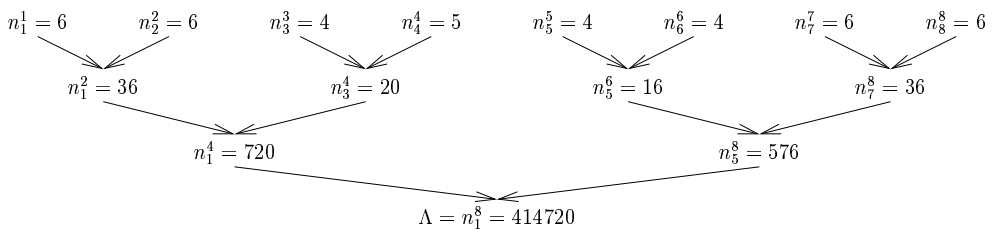
it appears that the complexity of processing the level l and the overall complexity are of the same order, i.e $O((K_H)^r)$. The same analysis applies to the algorithm 3. As an example, Fig. 2 illustrates the hierarchical algorithm with the same data as in table 2. Note that some values $n_{t_1}^{t_2}$ processed in Fig. 2 are not required (for example, the value $n_1^{K_H} = \Lambda$).

3.5 Compression efficiency

Each symbol of the sequence $\mathbf{S}_H$ is associated to a c-bits fixed length codeword, leading to a total length of the multiplexed flow equal to $c.K_H$. Therefore, the compression rate is determined by the amount of $\mathbf{S_L}$ information that can be jointly coded by the sequence of multiplexed codewords. Let us consider again the realization s_t of a symbol S_t of the sequence $\mathbf{S}_H$. One can associate an n_t-valued variable to the symbol value s_t. The amount of data that can be represented by this variable is $log_2(n_t)$ bits. Since the c bits of the multiplexed codeword code both the symbol value s_t and the index q_t describing the low priority bitstream realization, the mean description length (mdl) for the symbol s_t is theoretically $c - log_2(n_t) = -log_2(\frac{n_t}{2^c})$ bits. The amount of data, in bits, that may be multiplexed with a sequence $\mathbf{s}_H$ using the n_t-valued variables is given by

$$-\sum_{1 \le t \le K_H} log_2(\frac{n_t}{N}) \tag{13}$$

Processing of values $n_{t_1}^{t_2}$



Processing of values $q_{t_1}^{t_2}$

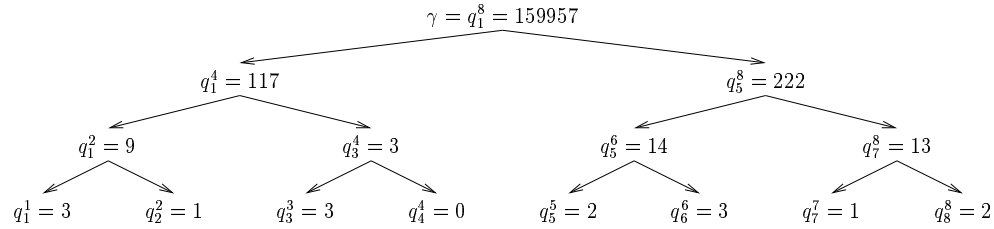


Figure 2: Hierarchical decomposition of γ.

In order to minimize the mdl $\hat{h}$ of the source $\mathbf{S}_H$, one has then to choose the partitioning $\mathcal{C}$ of the set of $N = 2^c$ fixed length codewords into *equivalence classes* $\mathcal{C}_i$ of cardinality $N_i, i = 1 \ldots \Omega$, such that

$$(N_1, N_2, \ldots, N_i, \ldots, N_\Omega) = arg\ min\left(-\sum_{i=1}^{\Omega} \mu_i\ log_2(\frac{N_i}{N})\right) = arg\ min(\hat{h}) \qquad (14)$$

In order to have a rate close to the entropy bound of the source $\mathbf{S}_H$, the quantity $\frac{N_i}{N}$ should be as close as possible to μ_i. We come back in section 5 on how to calculate an efficient set of N_i values for a given source pdf.

The entropy of the source $\mathbf{S}_H$ introduced in section 2 is $h = 1.658$. The partition $\mathcal{C}$ given in Table 1 leads to an mdl of $\mathbf{S}_H$ of $\hat{h} = 1.692$. Note that an Huffman encoder of the source would have led to an mdl equal to 1.840. The choice of c as well as of the partition $\mathcal{C}$ has an impact on the coding rate. For example, for $c = 6$, the partition into classes of respective cardinalities $(N_1, N_2, N_3, N_4) = (28, 19, 16, 1)$ produces an mdl equal to 1.662.

Property 1: $\forall\, \epsilon > 0, \exists\, c \in \mathbb{N}$ such that $\hat{h} - h \leq \epsilon$
In other words, the mdl of $\mathbf{S}_H$ can be made as close as required to the entropy, by increasing the length c of the codewords. The proof is provided in Appendix A.

4 A class of fast-computable multiplexed codes

The complexity of the previous algorithm, induced mainly by the Euclidean decomposition of the global variable γ, is $O({K_H}^2)$. In order to reduce the computational cost, we introduce in this section a class of constrained and fast-computable multiplexed codes. Thus, the pre-formatted lower priority bitstream is not seen as the representation of a unique variable γ, but as a sequence of "elementary variables". An elementary variable is defined as a variable which takes its value in a set of small dimension: it is a k-valued variable, k being small.

4.1 Constrained choice of partition $\mathcal{C}$

The N_i-valued variables, $1 \leq i \leq \Omega$, are decomposed into a set of elementary variables as follows:

$$N_i = \prod_{j=1}^{\nu_i} f_j^{\alpha_{i,j}} \tag{15}$$

where f_j are prime factors, for example $f_1 = 2, f_2 = 3, f_3 = 5, f_4 = 7, f_5 = 11, \ldots$ The term f_{ν_i} denotes the highest prime factor in the decomposition of N_i. The term $\alpha_{i,j}$ stands for the power of the prime factor f_j in the decomposition of N_i. In order to proceed with the expansion of the N_i-valued variables, the recursive Euclidean decomposition (cf. Eqn. (3) and (1)) of N_i by the sequence of its prime factor $(\underbrace{2, \ldots, 2}_{\alpha_{i,1}}, \ldots, \underbrace{f_{\nu_i}, \ldots, f_{\nu_i}}_{\alpha_{i,\nu_i}})$ is computed. This expansion of N_i-valued variables can be represented as:

$$N_i\text{-valued variable} \rightleftarrows \begin{cases} \alpha_{i,1} \text{ binary variables} \\ \alpha_{i,2} \text{ 3-valued variables} \\ \qquad\qquad \vdots \\ \alpha_{i,\nu_i} \ f_{\nu_i}\text{-valued variables} \end{cases} \tag{16}$$

Note that, if N_i is a prime factor, the previous decomposition is identity, and the algorithm has to proceed as in section 3.2. In order to limit the encoding complexity, the partitioning of the set of c-bits fixed length codes must be such that the set $(N_i)_{1 \leq i \leq \Omega}$ will not result in prime decompositions into factors greater than a given prime value f_ν, hence $N_i = \prod_{j=1}^{\nu} f_j^{\alpha_{i,j}}$. For example, for $c = 4$ and $f_\nu = 5$, each integer N_i must be chosen in the following set: 1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, 16. These additional constraints on the partition $\mathcal{C}$ will induce a loss in accuracy of the approximation of the pdf of the $\mathbf{S}_H$ source resulting in a higher mdl.

4.2 Conversion of the low priority bitstream into f_j-valued variables

Consecutive segments of the low priority bitstream are seen as the binary representations of integers to be decomposed this time into a set of $v_{\mathcal{T},1}$ binary, $v_{\mathcal{T},2}$ 3-valued, ..., $v_{\mathcal{T},j}$

t	1	2	3	4	5	6	7	8	
s_t	a_1	a_1	a_3	a_2	a_3	a_3	a_1	a_1	
n_t	6	6	4	5	4	4	6	6	
$\alpha_{t,1}$	1	1	2	0	2	2	1	1	$d_1 = 10$
$\alpha_{t,2}$	1	1	0	0	0	0	1	1	$d_2 = 4$
$\alpha_{t,3}$	0	0	0	1	0	0	0	0	$d_3 = 1$

Table 3: Calculation of the number of available f_j-valued variables.

bit 1	bit 2	bit 3	3-valued variable 1	3-valued variable 2
0	0	0	0	0
0	0	1	0	1
0	1	0	0	2
0	1	1	1	0
1	0	0	1	1
1	0	1	1	2
1	1	0	2	0
1	1	1	2	1
			2	2

Table 4: Example of transformation $\mathcal{T}$ ($u_{\mathcal{T}} = 3$).

j-valued, ..., $v_{\mathcal{T},\nu}$ f_ν-valued variables. This section introduces transformations that can be used for this conversion.

Let $\mathcal{T}$ be a transformation that takes $u_{\mathcal{T}}$ bits and produces a set of f_j-valued variables. Let $v_{\mathcal{T},j}$, $1 \leq j \leq \nu$, be the number of f_j-valued variables produced by a transformation $\mathcal{T}$. The number of possible values that can be taken by the set of f_j-valued variables resulting from the transformation $\mathcal{T}$ applied on $u_{\mathcal{T}}$ bits of the bitstream $\mathbf{b}$ is given by

$$\Lambda_{\mathcal{T}} = \prod_{j=1}^{\nu} f_j^{v_{\mathcal{T},j}}. \tag{17}$$

This $\Lambda_{\mathcal{T}}$-valued variable allows to theoretically store $log_2(\Lambda_{\mathcal{T}})$ bits. Note that $\Lambda_{\mathcal{T}}$ may be higher than $2^{u_{\mathcal{T}}}$. Therefore, there is an overhead per bit given by

$$o_{\mathcal{T}} = \frac{log_2(\Lambda_{\mathcal{T}})}{u_{\mathcal{T}}} - 1 \tag{18}$$

For example, the transformation $\mathcal{T}_{15}$ shown in table 4 applied on 3 bits produces two 3-valued variables, leading to a number of states increased from 8 to 9. On the decoder side, the extra state can not be observed if the transmission is error free, and can thus be used for error detection.

$\mathcal{T}_z$ identifier	$u_{\mathcal{T}_z}$	$v_{\mathcal{T}_z,1}$	$v_{\mathcal{T}_z,2}$	$v_{\mathcal{T}_z,3}$	$o_{\mathcal{T}_z}$
$\mathcal{T}_0$	1	1	0	0	0.0000
$\mathcal{T}_1$	15	0	8	1	0.0001
$\mathcal{T}_2$	21	0	3	7	0.0004
$\mathcal{T}_3$	19	0	12	0	0.0010
$\mathcal{T}_4$	25	0	7	6	0.0011
$\mathcal{T}_5$	24	0	2	9	0.0028
$\mathcal{T}_6$	14	0	3	4	0.0030
$\mathcal{T}_7$	18	0	7	3	0.0034
$\mathcal{T}_8$	27	0	1	11	0.0047
$\mathcal{T}_9$	17	0	2	6	0.0060
$\mathcal{T}_{10}$	30	0	0	13	0.0062
$\mathcal{T}_{11}$	20	0	1	8	0.0080
$\mathcal{T}_{12}$	11	0	7	0	0.0086
$\mathcal{T}_{13}$	23	0	0	10	0.0095
$\mathcal{T}_{14}$	6	0	1	2	0.0381
$\mathcal{T}_{15}$	3	0	2	0	0.0566
$\mathcal{T}_{16}$	2	0	0	1	0.1610
$\mathcal{T}_{z_{max}} = \mathcal{T}_{17}$	1	0	1	0	0.5850

Table 5: Transformations used for $f_\nu = 5$

Note that it is not useful to choose transformations with $v_{\mathcal{T},1} > 0$, since they transform binary variables into binary variables. Note also that the transformations as well as the parameter ν can be chosen such that $\Lambda_{\mathcal{T}} \leq 2^{32}$, to avoid arithmetic operations on high value integers.

Table 5 enumerates the transformations used for $f_\nu = 5$. If $f_\nu = 3$, only the transformations that do not use 5-valued variables are valid: $\mathcal{T}_0, \mathcal{T}_3, \mathcal{T}_{12}, \mathcal{T}_{15}, \mathcal{T}_{17}$. They are sorted by increasing loss in compression efficiency. Among all the possible sets $\mathbf{v}_{\mathcal{T}} = (v_{\mathcal{T},1}, v_{\mathcal{T},2}, \ldots, v_{\mathcal{T},j}, \ldots, v_{\mathcal{T},\nu})$ explored under previous constraints, only those for which the transformations introduce an overhead lower than 1% are kept. Increasing f_ν reduces this overhead.

4.3 Optimal set of transformations

Let again $\mathbf{s}_h$ be the realization of a source $\mathbf{S}_H$ mapped into a sequence of n_t-valued variables. Each n_t-valued variable is decomposed into a set of f_j-valued variables, each one being used $v_{\mathcal{T},j}$ times. Let d_j be the total number of f_j-valued variables involved in the expansion of the entire sequence of n_t-valued variables. The encoding proceeds with a projection of segments of the low priority bitstream into the set of f_j-valued variables resulting from the expansion of the n_t-valued variables associated to the realization $\mathbf{s}_h$ of the sequence of high priority symbols. Let $g_{\mathcal{T}_z}$ be the number of transformations $\mathcal{T}_z, z = 0, \ldots z_{max}$ necessary to convert the low priority bitstream into the f_j-valued variables. There are several possible

Algorithm 4 Choice of the transformations to be used

$z = 0$
while $sum(d_j) > 0$ **do** {while some f_j-valued variables are not computed yet}
  $g_{\mathcal{T}_z} = floor(min(\frac{d_j}{v_{\mathcal{T}_z,j}})$ {Calculation of how many transformations $\mathcal{T}_z$ can be used}
  for $\forall j$ between 1 and ν **do** {Update the number of f_j-variables to be transformed}
    $d_j = d_j - g_{\mathcal{T}_z} * v_{\mathcal{T}_z,j}$
    $z = z + 1$ {Try next transformation}
  end for
end while

sets $\mathbf{g} = (g_{\mathcal{T}_1} \dots g_{\mathcal{T}_z}, \dots g_{\mathcal{T}_{z_{max}}})$. The optimum set of transformations is the one which will minimize the total overhead, given by

$$O = \sum_{z=0}^{z_{max}} g_{\mathcal{T}_z} \; u_{\mathcal{T}_z} \; o_{\mathcal{T}_z} \tag{19}$$

The choice of $\mathbf{g}$ is an optimization problem. Knowing the transformation set $(\mathbf{T}_z)_{0 \leq z \leq z_{max}}$ and $\mathbf{d} = (d_1 \dots d_j \dots d_\nu)$, the optimal vector $\mathbf{g}$ is estimated by the algorithm 4.

4.4 Encoding procedure

Assuming that the low priority source s_L has been pre-encoded using efficient VLCs, the encoding of the two sequences s_H and s_L using fast multiplexed codes proceeds as follows:

1. For a given maximum value f_ν, the partition, hence the set of parameters (N_i), $i = 1, \dots, \Omega$, is calculated according to the criteria (14), under the constraint (15). For this purpose, the heuristic algorithm described in section 5 can be used.

2. For each prime integer f_j, $1 \leq j \leq \nu$, the number d_j of f_j-valued variables resulting from the expansion of the sequence of n_t-valued variables associated to s_H is calculated.

3. Consecutive segments of the low priority bitstream $\mathbf{b}$ are transformed into a sequence of f_j-valued variables. The number $g_{\mathcal{T}_z}$ of transformations $\mathcal{T}_z$ needed to perform the conversion is given by the procedure described in 4.3.

4. The transformations $\mathcal{T}_z$ are applied on consecutive segments of $u_{\mathcal{T}_z}$ bits of the bitstream $\mathbf{b}$, leading to the generation of sequences of f_j-valued variables.

5. The value q_t taken by the n_t-valued variable associated to the symbol realization s_t and to the realization of the low priority bitstream $\mathbf{b}$ is obtained by the euclidean multiplication of the f_j-valued variables calculated in the previous step.

6. The resulting pair (s_t, q_t) provides entries in the multiplexed codes table, leading to the selection of the multiplexed codes to be transmitted.

t	1	2	3	4	5	6	7	8
s_t	a_1	a_1	a_3	a_2	a_3	a_3	a_1	a_1
binary variables	1	0	10		10	11	0	0
3-valued variables	0	1					2	0
5-valued variables				1				
q_t	1	2	2	1	2	3	4	0
c_t	0001	0010	1101	0111	1101	1110	0100	0000

Table 6: Construction of the sequence of multiplexed codewords.

The decoding algorithm is made in the reverse order. As a consequence, it is not necessary to receive the whole sequence of codewords to decode the information of high priority $\mathbf{s_H}$: it is directly read in the multiplexed codes tables. Moreover, random access is also possible for the flow $\mathbf{s_H}$.

4.5 Example

Let us consider the sequence of 8 high priority symbols given in Table 3, with the corresponding mapping into the sequence of n_t-valued variables according to the multiplexed codes given in Table 1. The sequence of n_t-valued variables is decomposed into a sequence of d_1 binary, d_2 3-valued, and d_3 5-valued variables with respective powers $\alpha_{t,1}$, $\alpha_{t,2}$, and $\alpha_{t,3}$ (cf. table 3).

In a second step, one has to convert the low priority bitstream into the binary, 3-valued, and 5-valued variables. It appears that the transformations $\mathcal{T}_0$, $\mathcal{T}_{15}$ and $\mathcal{T}_{16}$ have to be used (see section 4.3) 10 times, 2 times and 1 time, respectively. Thus, 18 bits can be multiplexed with the realization $\mathbf{s_H}$ of $\mathbf{S_H}$, leading to a mdl of 1.75 bits per symbol. The transformation $\mathcal{T}_0$ being identity, the first 10 bits of the bitstream $\mathbf{b}$ are multiplexed unchanged. For each transformation $\mathcal{T}$, the $u_{\mathcal{T}}$ input bits are seen as the binary representation of an integer $\theta_{\mathcal{T}}$ as

$\mathcal{T}$		Input bits	$\rightarrow$	$\theta_{\mathcal{T}}$	$\rightarrow$	(3-valued variables,	5-valued variables)
$\mathcal{T}_{15}$	:	001	$\rightarrow$	1	$\rightarrow$	(01,	)
$\mathcal{T}_{15}$	:	110	$\rightarrow$	6	$\rightarrow$	(20,	)
$\mathcal{T}_{16}$	:	01	$\rightarrow$	1	$\rightarrow$	(,	1)

and leading to the sequences 1010101100 of binary variables, 0120 of 3-valued and 1 of 5-valued variables. The value q_t taken by the n_t-valued variable associated to the symbol realization s_t and to the realization of the low priority bitstream $\mathbf{b}$ is obtained by the projection of the segments of $\alpha_{i,j}$ bits of the sequences of binary ($j = 1$), 3-valued ($j = 2$) and 5-valued ($j = 3$) variables on the basis formed by the set of f_j-valued variables. The resulting pair (s_t, q_t) provides entries in the multiplexed codes table (see Table 1), leading to the selection of multiplexed codes c_t given in Table 6.

5 Probability distribution function approximation

As already mentioned above, the mdl $\hat{h}$ of the source $\mathbf{S}_H$ is function of the sizes N_i of the equivalence classes $\mathcal{C}_i, i = 1 \ldots \Omega$. This section describes an algorithm that aim to approximate, under constraints, the pdf of the source $\mathbf{S}_H$. The probability distribution function estimator $\hat{\boldsymbol{\mu}}$ is defined by

$$\hat{\boldsymbol{\mu}} = (\hat{\mu_1}, \hat{\mu_2}, \ldots, \hat{\mu_i}, \ldots, \hat{\mu_\Omega}) = \left(\frac{N_1}{N}, \frac{N_2}{N}, \ldots, \frac{N_i}{N}, \ldots, \frac{N_\Omega}{N} \right) \tag{20}$$

where $\sum_{i=1}^{\Omega} N_i \leq N = 2^c$. Let $\boldsymbol{\eta} = \{\eta_0 = 1, \eta_2 = 2, \ldots, \eta_k, \ldots, \eta_{max} = N\}$ be the set of possible values N_i. A higher number of valid values for N_i leads indeed to a better approximation of the pdf of the source $\mathbf{S}_H$. The probability μ_i of a symbol cannot be lower than $\frac{1}{N}$. The alphabet $\mathcal{A}$ is partitioned into two subsets $\mathcal{A}_m$ and $\mathcal{A}_M$ of sizes Ω_m and Ω_M, and defined respectively by

$$a_i \in \mathcal{A}_m \text{ iff } \mu_i < \frac{1}{N} \text{ and } a_i \in \mathcal{A}_M \text{ iff } \mu_i \geq \frac{1}{N} \tag{21}$$

Let $\tilde{\boldsymbol{\mu}}$ be the probability distribution function of symbols $a_i \in \mathcal{A}_M$ given by

$$\tilde{\mu_i} = \frac{\mu_i}{\sum_{a_i \in \mathcal{A}_M} \mu_i} \tag{22}$$

The algorithm aiming at the best partition of the set of codewords proceeds as follows:

1. For each $a_i \in \mathcal{A}_m$, N_i is set to 1.
2. The probability density function $\tilde{\boldsymbol{\mu}}$ of symbols belonging to $\mathcal{A}_M$ is calculated. Since Ω_m codewords have already been affected to classes of equivalence in the first step, there is still $N - \Omega_m$ codewords to be assigned to classes of equivalence. The expression (14) can then be written as

$$(N_1, N_2, \ldots, N_i, \ldots, N_\Omega) = arg\ min \left(log_2(N) \sum_{a_i \in \mathcal{A}_m} \mu_i - \sum_{a_i \in \mathcal{A}_M} \mu_i\ log_2(\frac{N_i}{N}) \right) \tag{23}$$

Since the first part of expression (23) is a constant and $N_i = 1$ when $a_i \in \mathcal{A}_m$, the optimization is performed for symbols of $\mathcal{A}_M$ only, and can be expressed as

$$\begin{aligned} (N_i)_{i \in \mathcal{A}_M} &= arg\ min \left(- \sum_{a_i \in \mathcal{A}_M} \mu_i\ log_2(\tfrac{N_i}{N}) \right) \\ &= arg\ min \left(- \sum_{a_i \in \mathcal{A}_M} \tfrac{\mu_i}{\sum_{i \in \mathcal{A}_M} \mu_i} \left(log_2(\tfrac{N_i}{N - \Omega_m}) + log_2(\tfrac{N - \Omega_m}{N}) \right) \right) \\ &= arg\ min \left(- \sum_{a_i \in \mathcal{A}_M} \tilde{\mu_i}\ log_2(\tfrac{N_i}{N - \Omega_m}) - \sum_{a_i \in \mathcal{A}_M} \tilde{\mu_i}\ log_2(\tfrac{N - \Omega_m}{N}) \right) \\ &= arg\ min \left(- \sum_{a_i \in \mathcal{A}_M} \tilde{\mu_i}\ \left(log_2(\tfrac{N_i}{N - \Omega_m}) \right) \right) \end{aligned} \tag{24}$$

As $\sum_{a_i \in \mathcal{A}_M} \tilde{\mu}_i = 1$, expression (24) can be seen as the expression of the mdl when the index i is constrained to be such that $a_i \in \mathcal{A}_M$. Consequently, assuming that $\frac{N_i}{N-\Omega_m}$ can take any real value, the optimum is given by:

$$\forall a_i \in \mathcal{A}_M, \ \frac{N_i}{N-\Omega_m} = \tilde{\mu}_i \tag{25}$$

In the following, the index i is supposed to be chosen such that $a_i \in \mathcal{A}_M$.

3. For each $a_i \in \mathcal{A}_M$, N_i is set to the highest value in $\boldsymbol{\eta}$ such that $N_i \leq \tilde{\mu_i}.(N-\Omega_m)$. The remaining number of codewords to be assigned to equivalence classes is then given by $\sigma = N - \sum_{i=1}^{\Omega} N_i \geq 0$.

4. As a consequence, it is possible to increase some values of N_i. For each index i between 1 and Ω, if the value η_{k+1} is used instead of η_k, the mdl $\hat{h}$ decreases of a positive value $\Gamma_i = log_2(\frac{\eta_{k+1}}{\eta_k})$. The decreasing mdl is induced by the assignment of $\eta_{k+1} - \eta_k$ pairs (a_i, N_i) to classes of equivalence and, in average per codeword, is given by $\gamma_i = \frac{\Gamma_i}{\eta_{k+1}-\eta_k}$. These pairs are then sorted by decreasing values of γ_i. The way to proceed is to choose the pair (a_i, N_i) with the highest associated value γ_i. For a given variable N_i set in the previous steps, it is possible to re-set it to η_{k+1} instead of η_k only if $\eta_{k+1} - \eta_k \leq \sigma$, i.e. if there is a sufficient number of codewords still to be assigned. If it is not the case, the pair (a_i, N_i) is ignored in the next iterations of the algorithm. The procedure continues with the treatment of the pair with the next value γ_i. If $\eta_{k+1} - \eta_k \leq \sigma$, then $N_i = \eta_{k+1}$, the value σ is set to $\sigma - \eta_{k+1} + \eta_k$, and the variable γ_i is updated for this symbol.

6 Error handling on a binary symmetric channel

In this section, we analyze the impact of channel errors on the distortion of the source $\mathbf{S_H}$. The impact on the sequence $\mathbf{b}$ is also discussed. The part of the flow $\mathbf{b}$ that is not multiplexed is supposed to be lost if an error occurs on the channel.

We consider a binary symmetric channel (BSC) with a bit error rate p. Let $q = 1-p$ be the probability that a bit is correctly received. The pdf $\boldsymbol{\mu}$ of $\mathbf{S_H}$ being known, it is possible to calculate the reconstruction accuracy in terms of Mean Square Error (MSE). Since each state q_t calculated from $\mathbf{b}$ can be seen as a uniform random variable, a symbol a_i of $\mathcal{A}$ has the same probability to be encoded with any codeword of its equivalence class $\mathcal{C}_i$. Hence,

$$P(c_{i,j}) = \frac{\mu_i}{N_i} \tag{26}$$

Let us consider a source represented by c bits fixed length codewords. The probability to receive the symbol y_n, if x_m has been transmitted, is function of the Hamming distance $h_{m,n}$ between the symbols y_n and x_m, and is given by

$$p_{m,n} = P(y_n/x_m) = p^{h_{m,n}} \ q^{c-h_{m,n}} \tag{27}$$

The MSE induced by the channel noise is then given by

$$\mathrm{MSE}_{fix} = \sum_{X_i \in \mathcal{X}} \mu_i \sum_{X_{i'} \in \mathcal{X}} p_{i,i'} \ \Delta(X_i, X_{i'}) \tag{28}$$

where $\Delta(X_i, X_{i'}) = \|X_i - X_{i'}\|^2$.

The reconstruction error of the source $\mathbf{S_H}$ encoded with multiplexed codes in presence of channel noise can be estimated similarly. Since the codewords c_i are FLC, Eqn. (27) still applies. The MSE of the reconstructed $\mathbf{S_H}$ stream in presence of bit errors can then be expressed as

$$\mathrm{MSE}_{mul,S_H} = \sum_{c_{i,j} \in \mathcal{C}} P(c_{i,j}) \sum_{c_{i',j'} \in \mathcal{C}} P(c_{i',j'}/c_{i,j}) \ \Delta'(c_{i,j}, c_{i',j'}) \tag{29}$$

where $P(c_{i',j'}/c_{i,j})$ stands for the probability to receive the codeword $c_{i',j'}$, if $c_{i,j}$ has been transmitted, and where $\Delta'(c_{i,j}, c_{i',j'})$ denotes the distortion of the $\mathbf{S_H}$ source induced by the reception of $c_{i',j'}$ instead of $c_{i,j}$. Let respectively a_i and $a_{i'}$ be the symbols decoded from $c_{i,j}$ and $c_{i',j'}$. Then $\Delta'(c_{i,j}, c_{i',j'}) = \Delta(a_i, a_{i'})$. This leads to

$$\mathrm{MSE}_{mul,S_H} = \sum_{a_i \in \mathcal{A}} \mu_i \sum_{a_{i'} \in \mathcal{A}} \Delta(a_i, a_{i'}) \underbrace{\frac{1}{N_i} \sum_{c_{i,j} \in \mathcal{C}_i} \sum_{c_{i',j'} \in \mathbf{C_{i'}}} P(c_{i',j'}/c_{i,j})}_{P(a_{i'}/a_i)}, \tag{30}$$

where $P(a_{i'}/a_i)$ is the probability to decode the symbol $a_{i'}$ if a_i has been transmitted. Therefore, the choice of the partition $\mathcal{C}$ has a major impact on the final distortion. However, the number of partitions for a given set of N_i-values is very high. For example, the number of partitions such that $N_1 = 6, N_2 = 5, N_3 = 4, N_4 = 1$, is 10 090 080.

Example: let us consider again the example of source alphabet given in section 3. Let

$$\Delta = \begin{bmatrix} 0 & 1 & 4 & 9 \\ 1 & 0 & 1 & 4 \\ 4 & 1 & 0 & 1 \\ 9 & 4 & 1 & 0 \end{bmatrix}$$

be the corresponding distortion matrix of elements $\Delta(a_i, a_{i'})$. The probabilities $P(a_{i'}/a_i)$ can be deduced from Eqn. 30. They are described by the transition transition matrix $\mathbf{P}_{\mathcal{A}}$, which is given by

$$\mathbf{P}_{\mathcal{A}} = \begin{bmatrix} \frac{6q^4+14pq^3+12p^2q^2+4p^3q}{6} & \frac{7pq^3+14p^2q^2+8p^3q+p^4}{6} & \frac{3pq^3+8p^2q^2+9p^3q+4p^4}{6} & \frac{2p^2q^2+3p^3q+p^4}{6} \\ \frac{7pq^3+14p^2q^2+8p^3q+p^4}{5} & \frac{5q^4+6pq^3+4p^2q^2+6p^3q+4p^4}{5} & \frac{6pq^3+9p^2q^2+5p^3q}{5} & \frac{pq^3+3p^2q^2+p^3q}{5} \\ \frac{3pq^3+8p^2q^2+9p^3q+4p^4}{4} & \frac{6pq^3+9p^2q^2+5p^3q}{4} & \frac{4q^4+4pq^3+6p^2q^2+2p^3q}{4} & \frac{3pq^3+p^2q^2}{4} \\ 2p^2q^2+3p^3q+p^4 & pq^3+3p^2q^2+p^3q & 3pq^3+p^2q^2 & q^4 \end{bmatrix}$$

For a BSC of BER equal to 0.1, the summation of Eqn. (30) leads to MSEs of 0.3686 and 0.4960 for respectively multiplexed codes and FLCs. Albeit quite surprising, this result is

engendered by the fact that the probabilities associated to symbols on which transmission errors will induce the largest distortion values (4 and 9) are lower for the multiplexed codes. The respective error resilience of FLCs and multiplexed codes hence depends on the statistical characteristics of the source $\mathbf{S_H}$. Cases where the channel induced distortion is higher with multiplexed codes than with FLCs may exist as well. However, practical sources generated by compression systems, can often be modeled as sources with generalized Gaussian pdf. For these sources, higher distortion values are induced by lower probability symbols, resulting in better performances for the multiplexed codes.

7 Discussion on Parameters choice

The choice of the parameters c and ν has a major impact on the performance, in terms of compression and error resilience, of the codes. Let h be the entropy per symbol of the source and $\hat{h}$ the mean codeword length produced by the algorithm. The overhead rate is given by $r_h = \frac{\hat{h}-h}{h}$. As c bits are always used to represent a symbol of the source $\mathbf{S_H}$, the rate of high priority data in the final multiplexed c-bits codewords stream is given by $\tau = \frac{h}{c}$.

We have seen above that c must be high enough so that one can find a partition, that will closely approximate the pdf of the source $\mathbf{S_H}$. On the other hand, a high value for the parameter c will result in a decreased rate of synchronous data. Thus a compromise has to be found between reaching the entropy (compression efficiency) and the rate of synchronous data in the multiplexed flow (error resilience). The ability to closely approximate the pdf of the source $\mathbf{S_H}$ (hence the closeness to entropy) depends also on the parameter ν. Fig. 3 shows the influence of the parameters c and ν on the rate r_h. For a source of cardinal $\Omega = 256$, the value $f_\nu = 5$ leads to a small overhead when the codeword length is $c = 14$. Notice that the number of available transformations and the number of flows of f_j-valued variables increase with ν. Therefore, the computational cost of the algorithm increases with ν.

8 Simulation results

To evaluate the performance of the multiplexed codes, experiments have been performed by considering a high priority source with a Gaussian distribution characterized by a zero-mean and a standard deviation $\sigma = 25$, and taking its values between -127 and $+127$. The source has been quantized with a scalar quantizer of step size 1. The entropy of this source is $h = 6.771$. We have considered a sequence $\mathbf{s_H}$ of 200 symbols. We assume that a lower priority sequence of symbols $\mathbf{s_L}$ has been pre-encoded into a bitstream $\mathbf{b}$ with a VLC coder. In the experiments reported here the sequence of bits $\mathbf{b}$ has been generated by a uniform random binary variable. Its length has been fixed to 2000. All the simulations have been performed assuming a binary symmetric channel with varying bit error rates (BER). The results are averaged over 100 channel realizations. Simulations have been performed considering respectively FLCs (8 bits), Huffman codes and lexicographical multiplexed codes. For the multiplexed codes, the parameters c and f_ν have been set to $c = 14$ and $f_\nu = 5$. The

mdl of the sequence $\mathbf{S_H}$ is 6.775. Fig. 4 shows the PSNR curves obtained for the sequence $\mathbf{S_H}$ with the three codes for varying channel BER. This figure reveals significantly higher performance for the multiplexed codes, even with respect to FLCs. The main reason is that the Hamming distance between symbols of "opposite" values is higher in the case of multiplexed codes: the Hamming distance is 14 for symbols -127 and $+127$, whereas it is only 8 in the case of FLCs. It has been observed that the average position of the first impaired bit in the Huffman bit stream is roughly the same. The reason is that the mean position of the first disturbed bit is statistically localized on the part of the low priority bitstream which is not multiplexed.

The multiplexed codes have then been experimented with real sources. We have considered a simple image coding system where the image is first decomposed into 16 subbands using a two-stage wavelet transform. The low and high frequency subbands have been quantized respectively on 7 and 4 bits. The high frequencies have been encoded with an Huffman algorithm, and then multiplexed into the low frequencies using multiplexed codes of parameters $c = 16$ and $f_\nu = 5$. A synchronization marker has been used every four lines for both Huffman codes and the low priority part of the multiplexed codes. The bit rates obtained with Huffman and multiplexed codes are quite similar and given respectively by 1.713 bit per pixel (*bpp*) and 1.712 *bpp*. When considering FLCs, the average bit rate obtained is 6.187 *bpp*. Fig. 5 depicts the PSNR and visual qualities obtained with the three codes. This figure evidences significant improvements both in PSNR and visual quality when using multiplexed codes even in comparison with FLCs and for a similar compression factor as the one obtained with Huffman codes.

9 Conclusion

We have introduced a new family of codes called "multiplexed codes". These codes avoid the "de-synchronization" phenomenon for the high priority source, while still allowing to reach asymptotically the entropy bound for both (low and high priority) sources. The low priority source can be pre-encoded with any efficient VLC. A FLC is created for the high priority source, its inherent redundancy being exploited to represent information from the low priority stream. Theory and simulations on both theoretical and real sources have evidence very high error resilience at almost no cost in terms of compression efficiency. Another key advantage is that they allow to make use of simple instantaneous decoding techniques. They hence appear to be excellent alternatives to reversible variable length codes (which suffer from some penalty in terms of compression efficiency, while not avoiding completely error propagation despite a decoding in two passes) or to classical VLCs for which robust decoding make often use of computationally expensive Bayesian estimation techniques.

References

[1] The gmp library. *http://www.swox.com/gmp/*.

[2] Anatoly A.Karatsuba and Y.Ofman. Multiplication of multidigit numbers on automata. *Soviet Physics Doklady*, January 1963. *http://cr.yp.to/bib/1963/karatsuba.html*.

[3] R. Bauer and J. Hagenauer. Turbo fec/vlc decoding and its application to text compressi on. In *Proc. Conf. on Inf. Theory and System*, pages WA6.6–WA6.11, March 2000.

[4] G. Cheung and A. Zakhor. Joint source-channel coding of scalable video over noisy channels. *Proc. of Intl. Conf. on Image Processing, ICIP*, pages 767–770, 1996.

[5] N. Demir and K. Sayood. Joint source-channel coding for variable length codes. In *Proc. IEEE Data Compression Conf., DCC*, pages 139–148, March 1998.

[6] A. Guyader, E. Fabre, C. Guillemot, and M. Robert. Joint source-channel turbo decoding of entropy coded sources. *IEEE Journal on Selected Areas in Communication, special issue on the turbo principle : from theory to practice*, 19(9):1680–1696, September 2001.

[7] J.P. Robinson J.C. Maxted. Error recovery for variables length codes. *IEEE Trans. on Inf. Theory*, IT-31(6):794–801, November 1985.

[8] W.M. Lam and A.R. Reibman. Self-synchronizing variable-length codes for image transmission. In *Proc. IEEE Intl. Conf. on Accoustic Speech and Signal Processing, ICASSP'92*, volume 3, pages 477–480, September 1992.

[9] A.H. Murad and T.E. Fuja. Joint source-channel decoding of variable length encoded sources. In *Proc. Inf. Theory Workshop, ITW*, pages 94–95, June 1998.

[10] M. J. Ruf and J. W. Modestino. Rate-distortion performance for joint source and channel coding of images. *Proc. of Intl. Conf. on Image Processing*, 2:77–80, October 1995.

[11] Y. Takishima, M. Wada, and H. Murakami. Reversible variable length codes. *IEEE Trans. On Communications*, 43(2/3/4):158–162, February 1995.

[12] N. Tanabe and N. Farvardin. Subband image coding using entropy-coded quantization over noisy channels. *IEEE Trans. on Selected Areas in Communications*, 10(5):926–942, June 1992.

[13] J.H. Rabinowitz T.J. Ferguson. Self-synchronizing huffman codes. *IEEE Trans. on Inf. Theory*, IT-30(4):687–693, July 1984.

[14] J. Wen and J. D. Villasenor. Reversible variable length codes for efficient and robust image and video coding. In *Proc. IEEE Data Compression Conf., DCC*, pages 471–480, April 1998.

Appendix A: Proof of property 1

Let $\epsilon \in \mathbb{R} - \{0\}$. The problem consists in finding a partition $\mathcal{C}$ that verifies the property. Let c_h be the integer defined as $c_h = \lceil -log_2(\min_{i \in \mathcal{A}}(\mu_i)) \rceil$. By definition,

$$\forall\, i \in \mathcal{A},\ \mu_i \geq 2^{-c_h} \tag{31}$$

For all $c \in \mathbb{N}$ such that $c \geq c_h$, we choose $N_i = \lfloor \mu_i\, 2^c \rfloor$. From $\sum_{i \in \mathcal{A}} \mu_i = 1$ we get $\sum_{i \in \mathcal{A}} \lfloor \mu_i\, 2^c \rfloor \leq 2^c \sum_{i \in \mathcal{A}} \mu_i = 2^c$. Moreover, $\forall i,\ N_i \geq 1$. Therefore this choice of $(N_i)_{1 \leq i \leq \Omega}$ is valid. By construction, $\mu_i\, 2^c - 1 \leq N_i \leq \mu_i\, 2^c$, hence,

$$1 \leq \mu_i \frac{2^c}{N_i} \leq 1 + \frac{1}{\mu_i\, 2^c - 1} \tag{32}$$

The difference δ_h between the mdl and the entropy is given by

$$\delta_h = \hat{h} - h = -\sum_{i \in \mathcal{A}} \mu_i\ log_2(\frac{N_i}{2^c}) + \sum_{i \in \mathcal{A}} \mu_i\ log_2(\mu_i) = \sum_{i \in \mathcal{A}} \mu_i\ log_2(\frac{\mu_i\, 2^c}{N_i}) \tag{33}$$

therefore, from Eqn (32), it can be seen easily that δ_h verifies

$$\begin{array}{rrl} & \delta_h & \leq \sum_{i \in \mathcal{A}} \mu_i\ log_2(1 + \frac{1}{\mu_i\, 2^c - 1}) \\ \Rightarrow & \delta_h & \leq \sum_{i \in \mathcal{A}} \frac{\mu_i}{\mu_i\, 2^c - 1} = \sum_{i \in \mathcal{A}} \frac{1}{2^c - \frac{1}{\mu_i}}. \end{array} \tag{34}$$

From Eqn. (31) and (34) we get

$$\delta_h \leq \sum_{i \in \mathcal{A}} \frac{1}{2^c - 2^{c_h}} = \frac{\Omega}{2^c - 2^{c_h}} \tag{35}$$

Thus, using $c = \lceil log_2(\frac{\Omega}{\epsilon} + 2^{c_h}) \rceil$ and $\forall i,\ N_i = \lfloor \mu_i\, 2^c \rfloor$, the inequality $\delta_h \leq \epsilon$ is verified.

Notice that, for any code for which the inequality $c \geq c_h$ is true, this proof provides an upper bound for the mdl. □

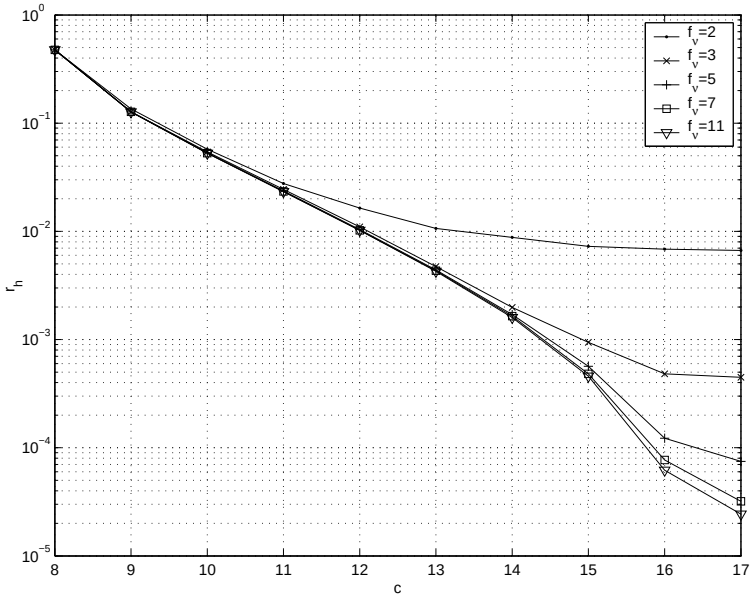


Figure 3: Closeness to entropy of fast-computable Multiplexed codes (Gaussian source).

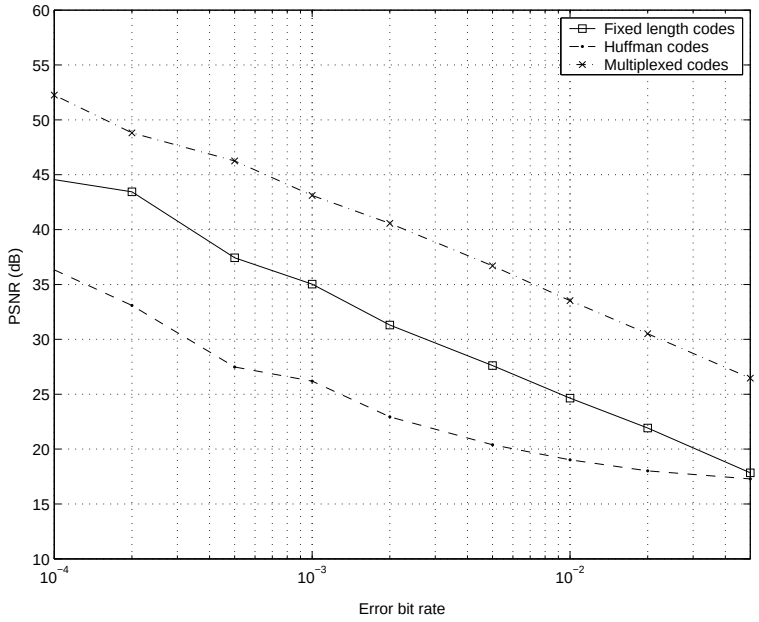


Figure 4: PSNR of the source $\mathbf{S_H}$ (Gaussian distribution)

Figure 5: PSNR performance and visual quality obtained respectively with FLCs, Huffman codes and multiplexed codes. The channel bit error rates are 0.0005 (top images), 0.005 (middle images) and 0.05 (bottom images).

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399