On the Complexity of Joint Source-Channel Decoding of Markov Sequences over Memoryless Channels

Sorina Dumitrescu and Xiaolin Wu
McMaster University, Hamilton, Ontario, Canada
Email: sorina/xwu@mail.ece.mcmaster.ca

Abstract

We investigate the complexity of joint source-channel maximum a posteriori (MAP) decoding of a Markov sequence which is first encoded by a source code, then encoded by a convolutional code, and sent through a noisy memoryless channel. As established previously the MAP decoding can be performed by a Viterbi-like algorithm on a trellis whose states are triples of the states of the Markov source, source coder and convolutional coder. The large size of the product space (in the order of K^2N, where K is the number of source symbols and N is the number of states of the convolutional coder) appears to prohibit such a scheme. We show that for finite impulse response convolutional codes, the state space size of joint source-channel decoding can be reduced to $O(K^2 + N \log N)$, hence the decoding time becomes $O(TK^2 + TN \log N)$, where T is the length in bits of the decoded bitstream. We further prove that an additional complexity reduction can be achieved when $K > N$, if the logarithm of the source transition probabilities satisfy the so-called Monge property. This decrease becomes more significant as the tree structure of the source code is more unbalanced. The reduction factor ranges between $O(K/N)$ (for a fixed-length source code) and $O(K/\log N)$ (for Golomb-Rice code).

This work was presented in part at the 2005 IEEE International Information Theory Symposium, Adelaide, Sept. 2005.

Key words: Joint source-channel decoding, maximum a posteriori sequence estimation, finite-state machine, complexity.

I. Introduction

In traditional communications systems for noisy channels the source coder and channel coder are designed independently. This practice is guided by Shannon's famous source-channel separation theorem: Optimum performance can be achieved asymptotically in the code block length and complexity by carrying out the tasks of compression and error correction separately. However, in practice, the design of such a perfect separation system has to respect delay and complexity constraints, thus leading to suboptimality. Moreover, Shannon's separation theorem no longer holds in network-based communications.

One way to improve the performance of practical coding systems, which has attracted considerable attention of researchers lately, is joint source-channel decoding. When a suboptimal design of a source coder fails to completely remove the redundancy of the source, some residual redundancy will be present at the input of the channel coder ([14]). Joint source-channel decoding will use this residual source redundancy solely or in conjunction with additional redundancy provided by a channel coder to correct transmission errors.

The source encoding of a continuous correlated source is usually performed in two steps. First the source is quantized and then each quantizer output is assigned a source codeword. Due to the exponential explosion of the quantizer complexity as the sample block size increases, only modest block size can be used in practice. But for a highly correlated source the block size of a practical quantization scheme may not be large enough to completely remove the correlation. Therefore, memory will be present in the quantizer output that may be modeled as a discrete Markov source.

Many researchers have assumed such a model and investigated joint source-channel decoding of discrete Markov sequences, which are encoded symbol by symbol, followed or not by channel coding, and sent through a noisy channel. A popular technique for joint source-channel decoding in such systems is maximum a posteriori (MAP) Markov sequence estimation. In [11], [1], [12] MAP Markov sequence decoders were proposed for the case when the source codewords have fixed-length and no channel code is applied. In [13], [17], [20] the case of variable-length source code was treated. MAP Markov sequence decoders for systems where the source code is followed by a channel code were considered in [4], [10] (convolutional code), and [6] (turbo code). Iterative decoders for systems where the source coder is followed by an interleaver and then a convolutional coder were also investigated [7], [8]. End-to-end minimum mean squared distortion decoders were studied in [18], [19].

This paper reexamines joint source-channel decoding by MAP sequence estimation in the case where a Markov sequence (MS) is first compressed by a source coder (SC), then protected by a convolutional coder (CC), and sent through a memoryless noisy channel. The problem has been studied previously in [10], [7], [4]. The main observation in constructing such a MAP decoder is that the cascaded chain formed by the MS followed by the SC and the CC (MS$\rightarrow$SC$\rightarrow$CC) can be modelled as a finite-state machine whose states are triples of the states of the three elements in the chain. Thus a trellis can be constructed to perform a Viterbi-like decoding. The complexity of decoding depends on the state space size. Since the number of all possible triples is $O(K^2N)$, where K is the number of source symbols and N is the number of states of the CC, the decoding appears to be prohibitively expensive [7]. It was suggested in [4], [10] that the number of states of the combined state machine can be reduced, but the significance of this reduction was only assessed on some simple examples.

We focus on the complexity of the joint source-channel MAP sequence decoder in the case of a finite impulse response (FIR) convolutional code. An FIR convolutional encoder can be realized in controller canonical form, in which the current state identifies the sequence of $m = \log_2 N$ previously encoded information bits. Based on this observation, we show that the space size of the joint source-channel decoder can be reduced to less than $K^2 + N \log N$. This leads to a decoding time complexity of $O(TK^2 + TN \log N)$, where T is the length in bits of the received bitstream. Additionally, we prove that the decoding space complexity is only $O(TK + TN)$. Although we concentrate in this paper on evaluating the complexity of the MAP sequence decoder, we mention that our result on the state space size reduction also applies to other trellis-based decoding algorithms like symbol-by-symbol MAP decoding or end-to-end minimum mean squared error decoding.

Moreover, we prove that if the logarithm of the source transition probabilities satisfies the so-called Monge property, the decoding complexity can be further reduced, if $K > N$, by applying a fast matrix search technique [2]. The reduction depends on the tree structure of the source coder and is larger if the tree is more unbalanced. The reduction factor ranges between $O(K/N)$

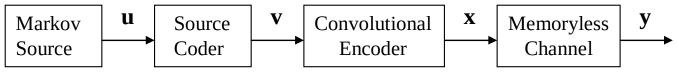

Fig. 1. The sequential operations of the system.

(for a fixed-length source code) and $O(K/\log N)$ (for Golomb-Rice code).

The paper is structured as follows. The next section formulates the problem. Section 3 presents a combined finite-state machine (CFSM) of the Markov model, source coder and convolutional encoder. In Section 4 we show how the state-space size of the CFSM can be reduced and analytically evaluate the reduction. The following section describes the MAP joint source-channel decoding algorithm based on the trellis constructed from the reduced CFSM. In Section 6 we propose an alternative decoder graph and in Section 7 we show how to use the structure of this graph in conjunction with a fast matrix search technique to accelerate the MAP decoding, if the logarithm of the source transition probabilities satisfies the so-called Monge property. Section 8 concludes the paper.

II. Problem Formulation

We consider a first order Markov source (MS) over an alphabet of K symbols $\{a_1, a_2, \cdots, a_K\}$, with conditional probabilities $P(a_i|a_j)$, $1 \leq i, j \leq K$, and initial probabilities $P(a_i)$, $1 \leq i \leq K$. Let $\mathbf{u} = u_1 u_2 \cdots u_I$ be a sequence generated by the MS. The Markov sequence $\mathbf{u}$ is coded by a source code (SC), which can be a fixed or a variable-length code, into the bitstream $\mathbf{v} = v_1 v_2 \cdots v_M$, $v_j \in \{0, 1\}$. Further a convolutional encoder (CC) is applied to the sequence $\mathbf{v}$, generating a new bit sequence $\mathbf{x} = x_1 x_2 \cdots x_T$, $x_k \in \{0, 1\}$, which is sent through a noisy memoryless channel. The bit sequence received at the other end is denoted by $\mathbf{y} = y_1 y_2 \cdots y_T$.

The problem of joint source-channel maximum a posteriori probability (MAP) sequence decoding is, given the sequence $\mathbf{y} = y_1 y_2 \cdots y_T$ output by the noisy channel, to infer the input Markov sequence $\mathbf{u} = u_1 u_2 \cdots u_I$ of maximal a posteriori probability $P(\mathbf{u}|\mathbf{y})$. The whole process is shown in Figure 1.

Since the bitstream $\mathbf{x}$ is completely determined by the Markov sequence $\mathbf{u}$ and vice versa, $P(\mathbf{y}|\mathbf{u})$ equals the channel transition probability $P_e(\mathbf{y}|\mathbf{x})$. Because the channel is memoryless, we have $P_e(\mathbf{y}|\mathbf{x}) = \Pi_{i=1}^{T} P_e(y_i|x_i)$. Since $\mathbf{u}$ is generated by a first order MS, we have $P(\mathbf{u}) =$

$P(u_1)\Pi_{i=2}^{I}P(u_i|u_{i-1})$. It follows from Bayes' Theorem that the MAP sequence decoding problem is equivalent to finding the Markov sequence $\mathbf{u} = u_1u_2\cdots u_I$ and/or the corresponding bitstream $\mathbf{x} = x_1x_2\cdots x_T$ obtained by cascading the SC and CC, such that the product

$$P(u_1)\Pi_{i=2}^{I}P(u_i|u_{i-1})\Pi_{i=1}^{T}P_e(y_i|x_i) \tag{1}$$

is maximized, given the bitstream $\mathbf{y} = y_1y_2\cdots y_T$ output by the noisy channel. Note that T is known at the decoder, while I is known only if a fixed-length SC is applied. Otherwise(i.e., in the case of a variable-length SC) I is a variable in the optimization.

III. Combined Finite-state Machine of Markov Model, Source Code and Channel Code

As noted in [10], [7], [4], the generation of the bitstream $\mathbf{x}$ via the composite actions of MS, SC and CC, can be modelled by a stochastic finite-state machine, which we will call the combined finite-state machine (CFSM).

The MS is conventionally modelled by a finite-state machine whose states are identified with the alphabet symbols. There is a transition from a state a_j to any state a_i, which outputs the symbol a_i, with probability $P(a_i|a_j)$. There is a distinct state a_0, the initial state. There are transitions from a_0 to any other state a_i, with output a_i, and probability $P(a_i)$.

The SC applied to the Markov sequence can be a fixed-length code or a variable-length code which is prefix free. Let c_i denote the binary codeword assigned to the source symbol a_i, $1 \leq i \leq K$. Since the code is prefix-free, it has a binary tree representation. The tree has K leaves and $K-1$ internal nodes. Let L_1 denote the root and $L_2, \cdots, L_{K-1}$ denote the other internal nodes. For convenience we will refer to the bit sequence which labels the path from the root to this node as bit sequence L_j. By $|L_j|$ we denote the number of bits in this sequence. Note that L_1 corresponds to the empty sequence, hence $|L_1| = 0$.

The bitstream generated by the MS and the SC (MS$\rightarrow$SC) can be described as the output of a finite-state machine (FSM) in the following way. The states are all pairs (a_i, L_j), $0 \leq i \leq K$, $1 \leq j \leq K-1$. The initial state is (a_0, L_1). Since there is a one-to-one correspondence between alphabet symbols and codewords, we will identify any state (a_i, L_j) with the pair (c_i, L_j), with the convention that c_0 is the empty word. Before describing the transitions of this FSM, let us explain first the meaning of each state. If after a number of transitions the system is in state

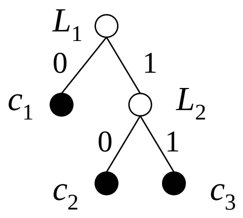

Fig. 2. The source coder tree of Example 1. The filled circles represent the leaves, i.e. the source codewords, and the empty circles represent the internal nodes, i.e. the prefixes of codewords.

	(c_1, L_1)	(c_2, L_1)	(c_3, L_1)	(c_1, L_2)	(c_2, L_2)	(c_3, L_2)
(c_1, L_1)	0			1		
(c_2, L_1)	0				1	
(c_3, L_1)	0					1
(c_1, L_2)		0	1			
(c_2, L_2)		0	1			
(c_3, L_2)		0	1			

TABLE I
TRANSITION MATRIX IN THE FSM OF THE MS$\to$SC PROCESS OF EXAMPLE 1. THE ELEMENTS 0 AND 1 MARK VALID TRANSITIONS WITH OUTPUT 0 AND 1, RESPECTIVELY.

(c_i, L_j), then the last completed codeword in the output sequence is c_i, which is followed by the bit sequence L_j. From each state (c_i, L_j) there are only two possible transitions, one outputting the bit 0 and the other outputting bit 1. The transition outputting the bit b either ends at the state $(c_i, L_{j'})$ if appending the bit b to the end of L_j generates the bit sequence $L_{j'}$, or at the state $(c_{i'}, L_1)$ if appending the bit b to the end of L_j generates the codeword $c_{i'}$.

Let the convolutional encoder (CC) which is next applied, be a feedforward convolutional encoder of rate $1/\beta$, with memory order m, realized in controller canonical form. Our terminology used in describing the CC is from [9]. The case of higher rate CC will be briefly discussed later.

The CC consists of one shift register of length m. The input sequence enters the left end of the shift register and the β output sequences are produced by β modulo 2 adders external to the shift register. The β output sequences are interleaved to produce the channel codeword.

The CC itself can be described by a state transition diagram. The state of the encoder is defined as its shift register contents. Thus, the total number of states of the CC is $N = 2^m$,

	S_1	S_2	S_3	S_4
S_1	0/000	1/111		
S_2			0/011	1/100
S_3	0/001	1/110		
S_4			0/010	1/101

TABLE II

TRANSITION MATRIX OF THE CC OF EXAMPLE 1. THE ENTRIES $b/b'_1b'_2b'_3$ MARK VALID TRANSITIONS WITH INPUT b AND OUTPUT $b'_1b'_2b'_3$.

corresponding to all m-bit sequences. We denote the states by $S_1, \cdots S_N$, S_1 being the initial state. For each $k, 1 \leq k \leq N$, let the sequence of m bits $b_1^k b_2^k \cdots b_m^k$ be the memory content of state S_k. Unlike the conventional notation, we assume that b_1^k is the rightmost bit in the shift register and b_m^k is the leftmost. Because the convolutional encoder is a feedforward encoder in controller canonical form, the contents of its shift register are its previous m input bits. Hence the CC is in state $b_1^k b_2^k \cdots b_m^k$ at some time t if and only if the input processed at time $t - i$ is b_{m+1-i}^k, for all $i, 1 \leq i \leq m$. After the current information bit b is processed, the new state of the CC is $b_2^k \cdots b_m^k b$. Thus, there are two transitions starting from any state in the state diagram of the CC, one with input bit 0, the other with input bit 1. A transition with input bit b is from any state S_k to any state $S_{k'}$ such that $b_1^{k'} b_2^{k'} \cdots b_m^{k'} = b_2^k \cdots b_m^k b$. Every transition outputs a sequence of β bits according to the generator matrix of the convolutional encoder.

Example 1. Consider a Markov source over a three-symbol alphabet, the SC with codewords $c_1 = 0$, $c_2 = 10$, $c_3 = 11$, and a rate $1/3$ CC with memory order $m = 2$. The tree representation of the SC is depicted in Figure 2. The transition matrix of the FSM of the MS$\rightarrow$SC process is given by Table I. Table II shows the transition matrix of the CC. Note that the states of CC are: $S_1 = 00$, $S_2 = 01$, $S_3 = 10$, $S_4 = 11$.

The states of the combined finite-state machine (CFSM) which describes the overall system of MS$\rightarrow$SC$\rightarrow$CC, are all the triples (c_i, L_j, S_k), $0 \leq i \leq K$, $1 \leq j \leq K-1$, $1 \leq k \leq N$. The initial state is (c_0, L_1, S_1). There is a transition from some state (c_i, L_j, S_k) to a state $(c_{i'}, L_{j'}, S_{k'})$ if and only if there is a transition with output $b \in \{0, 1\}$ from the state (c_i, L_j) to the state $(c_{i'}, L_{j'})$ in the FSM of MS$\rightarrow$SC, and there is a transition with input bit b from the state S_k to state $S_{k'}$

in the state diagram of the CC. Note that in this case the bit b is unique. The output of the transition from (c_i, L_j, S_k) to $(c_{i'}, L_{j'}, S_{k'})$ is the output of the transition from S_k to $S_{k'}$ in the state diagram of the CC.

IV. State Reduction in CFSM

The total number of states of the CFSM, if constructed mechanically as described in the preceding section, is clearly $(K+1)(K-1)N$. This seemingly suggests that the joint source-channel MAP decoding would have a complexity of at least $O(K^2NT)$, where T is the length of the sequence to be decoded in bits. Indeed, in [7] the authors made remarks about the prohibitive cost of the joint source-channel MAP decoding. But an observation lending hope for complexity reduction is that many of the triples (c_i, L_j, S_k) can never be reached by any sequence of transitions starting from the initial state (c_0, L_1, S_1), hence they can be eliminated from the CFSM. This observation was made in [10] and [4], but it was not evaluated how significant this reduction can be, except on some simple cases. Next we will answer this question precisely.

A state (c_i, L_j, S_k) of the CFSM can be reached through a sequence of transitions from the initial state if there is a bitstream $v_1v_2\cdots v_n$, a sequence of transitions from (c_0, L_1) to (c_i, L_j) which outputs the bitstream $v_1v_2\cdots v_n$ in the finite-state machine of MS$\rightarrow$SC, and a sequence of transitions from S_1 to S_k with input $v_1v_2\cdots v_n$ in the CC state diagram. Then c_iL_j has to be a suffix of $v_1v_2\cdots v_n$, and $b_1^kb_2^k\cdots b_m^k$ has to be a suffix of $v_1v_2\cdots v_n$, too. It follows that if $m \leq |c_i| + |L_j|$ then $b_1^kb_2^k\cdots b_m^k$ is a suffix of c_iL_j, and if $m > |c_i| + |L_j|$ then c_iL_j is a suffix of $b_1^kb_2^k\cdots b_m^k$. By eliminating all the triples which do not satisfy the above property a reduced combined finite-state machine (RCFSM) is obtained. Table III presents the transition matrix of the RCFSM for Example 1.

The decoding complexity is determined by the number of states (c_i, L_j, S_k) for $i \geq 1$, called *significant states*, in the RCFSM. In Table IV the significant states of the RCFSM for Example 1 are listed. Denote by $\mathcal{S}$ the set of significant states in RCFSM. To evaluate $|\mathcal{S}|$ we partition $\mathcal{S}$ into two subsets $\mathcal{C}_I$ and $\mathcal{C}_{II}$. $\mathcal{C}_I$ contains the states with $m \leq |c_i| + |L_j|$, while $\mathcal{C}_{II}$ contains those with $m > |c_i| + |L_j|$. For each pair (c_i, L_j) such that $m \leq |c_i| + |L_j|$ there is only one state S_k of the convolutional coder such that $b_1^kb_2^k\cdots b_m^k$ is a suffix of c_iL_j. Consequently,

$$|\mathcal{C}_I| = \sum_{i,j,|c_i|+|L_j|\geq m} 1 \leq K(K-1). \tag{2}$$

	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7
$Q_1 = (c_1, L_1, S_1)$	000		111				
$Q_2 = (c_1, L_1, S_3)$	001		110				
$Q_3 = (c_1, L_2, S_2)$				011		100	
$Q_4 = (c_2, L_1, S_3)$	001				110		
$Q_5 = (c_2, L_2, S_2)$				011		100	
$Q_6 = (c_3, L_1, S_4)$		010					101
$Q_7 = (c_3, L_2, S_4)$				010		101	

TABLE III

TRANSITION MATRIX FOR THE RCFSM OF EXAMPLE 1. THE THREE-BIT SEQUENCES ARE THE OUTPUTS OF VALID TRANSITIONS.

	S_1	S_2	S_3	S_4
(c_1, L_1)	$\times$		$\times$	
(c_2, L_1)			$\times$	
(c_3, L_1)				$\times$
(c_1, L_2)		$\times$		
(c_2, L_2)		$\times$		
(c_3, L_2)				$\times$

TABLE IV

SIGNIFICANT STATES OF THE RCFSM OF EXAMPLE 1.

Since the states of the CC correspond to all sequences of m bits, for each pair (c_i, L_j) such that $m > |c_i| + |L_j|$ there are $2^{m-|c_i|-|L_j|}$ states S_k of the convolutional coder such that $c_i L_j$ is a suffix of $b_1^k b_2^k \cdots b_m^k$. Therefore,

$$\begin{aligned} |\mathcal{C}_{II}| &= \sum_{i,j,|c_i|+|L_j|<m} 2^{m-|c_i|-|L_j|} \\ &= 2^m \sum_{i,j,|c_i|+|L_j|<m} 2^{-|c_i|-|L_j|} \\ &\leq 2^m \sum_{j,|L_j|<m} 2^{-|L_j|} \sum_{i=1}^{K} 2^{-|c_i|}. \end{aligned} \tag{3}$$

According to Kraft's inequality $\sum_{i=1}^{K} 2^{-|c_i|} \leq 1$, we have

$$|\mathcal{C}_{II}| \leq 2^m \sum_{j,|L_j|<m} 2^{-|L_j|}. \tag{4}$$

On the other hand

$$\sum_{j,|L_j|<m} 2^{-|L_j|} = \sum_{l=0}^{m-1} \sum_{j,|L_j|=l} 2^{-l} \leq \sum_{l=0}^{m-1} 1 = m. \tag{5}$$

We used above the inequality $\sum_{j,|L_j|=l} 2^{-l} \leq 1$, which follows from the fact that there are at most 2^l nodes L_j on the l-th level of the source coder tree. Relations (4) and (5) imply that

$$|\mathcal{C}_{II}| \leq m 2^m = N \log_2 N. \tag{6}$$

Finally,

$$|\mathcal{S}| = |\mathcal{C}_I| + |\mathcal{C}_{II}| \leq K^2 - K + N \log_2 N. \tag{7}$$

Note that this upper bound is very loose actually. Indeed, the equality in (2) holds only if $\mathcal{C}_{II}$ is empty, and the equality in (6) holds only if $\mathcal{C}_I$ is empty. Clearly these two conditions cannot hold simultaneously. The precise count of $|\mathcal{S}|$ depends on how balanced the source code tree is. We consider two extreme cases to illustrate the point: the Golomb-Rice code (the worst ill-balanced code tree) and the fixed-length code (perfectly balanced code tree). The detailed derivations of these results are deferred to Appendix.

For the Golomb-Rice code, each codeword c_i, for $1 \leq i \leq K-1$, is the bit sequence of $i-1$ 1's followed by a 0, and c_K is the bit sequence of $K-1$ 1's. For each $j, 2 \leq j \leq K-1$, L_j is the bit sequence of $j-1$ 1's. We assume that $m \leq K-1$, which is often the case in practice. Then the following relation holds

$$|\mathcal{S}| = K^2 - K + 2N - \tfrac{1}{2}(\log_2^2 N + 3\log_2 N + 4), \tag{8}$$

which implies that $|\mathcal{S}| = O(K^2 + N)$.

For a fixed-length source code with K being an integer power of 2, the following statements are valid:

i) if $N \leq K$ then $|\mathcal{S}| = K^2 - K$;

ii) if $K < N \leq K^2/2$ then

$$|\mathcal{S}| = K^2 + N \log_2 N - N(1 + \log_2 K); \tag{9}$$

iii) if $N > K^2/2$ then $|\mathcal{S}| = N \log_2 K$.

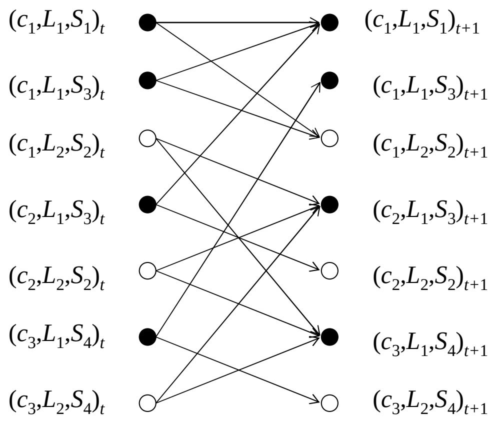

Fig. 3. Decoder graph: two consecutive stages and edges between them; black circles are nodes corresponding to the completion of a source codeword.

V. Decoding Algorithm

Having analyzed the structure of the RCFSM, we are now ready to present a MAP decoding algorithm and evaluate its time and space complexity.

In order to determine an input Markov sequence $\mathbf{u}$, it suffices to find the corresponding sequence of transitions of the RCFSM. If the channel output sequence $\mathbf{y}$ has T bits, then the channel input sequence $\mathbf{x}$ also has T bits. Hence $\mathbf{x}$ is produced after $M = T/\beta$ transitions in the RCFSM. The MAP sequence decoding is performed on a so-called decoder graph G on RCFSM. G is a trellis of $M+1$ stages, numbered from 0 to M, each stage containing as nodes the states of the RCFSM, in general. The nodes situated at stage t, $0 \leq t \leq M$, are denoted by $(c_i, L_j, S_k)_t$ for all i, j, k such that (c_i, L_j, S_k) is a state of the RCFSM.

There are edges (transitions) between any two nodes $(c_i, L_j, S_k)_t$ and $(c_{i'}, L_{j'}, S_{k'})_{t+1}$ such that there is a transition between the states (c_i, L_j, S_k) and $(c_{i'}, L_{j'}, S_{k'})$ in the RCFSM (Figure 3). Such an edge corresponds to the decoding of the subsequence $y_{t\beta+1} \cdots y_{t\beta+\beta}$ of $\mathbf{y}$, into the bitstream $b_1 b_2 \cdots b_\beta$, which is the output of the transition from (c_i, L_j, S_k) to $(c_{i'}, L_{j'}, S_{k'})$ in the RCFSM. The weight of this edge should in principle have two components: one accounting for the channel error probability and the other for the MS transition probability. However it is enough to assign a component accounting for the MS probability only if the edge ends at a complete node, i.e. which corresponds to the completion of a source codeword, or equivalently, $j' = 1$. Therefore, we define the weight of the edge from the node $(c_i, L_j, S_k)_t$ to the node

$(c_{i'}, L_{j'}, S_{k'})_{t+1}$ to be $\log P_e(y_{t\beta+1} \cdots y_{t\beta+\beta} | b_1 b_2 \cdots b_\beta) + \log P(a_{i'}|a_i)$, if $j' = 1$, and to be $\log P_e(y_{t\beta+1} \cdots y_{t\beta+\beta} | b_1 b_2 \cdots b_\beta)$, otherwise. This distinction is justified by the following observation. Any path connecting two complete nodes, but whose internal nodes are all incomplete, corresponds to an MS transition. Then the probability of this transition is accounted for in the weight of the path only through the last edge. Consider the path between two nodes $(c_i, L_1, S_k)_t$ and $(c_{i'}, L_{j'}, S_{k'})_{t+|c_{i'}|}$ (note that the path connecting these nodes is unique). It corresponds to the MS transition from state a_i to state $a_{i'}$. On the other hand, summing the weights of the component edges obtains the weight of the path to be $\log P(a_{i'}|a_i) + \mathcal{C}$, where $\mathcal{C}$ depends only on the channel error probability.

The final nodes of the graph are $(c_i, L_1, S_k)_M$, for all $i \geq 1$, each corresponding to the completion of a codeword. There is a one-to-one correspondence between a path from the initial node to some final node and a pair $\mathbf{u}, \mathbf{x}$, such that the weight of the path equals the logarithm of (1). Thus, the problem of joint source-channel MAP sequence decoding is reduced to the maximum-weight path problem in G.

The MAP sequence decoding algorithm proceeds stage by stage and computes the maximum-weight path ending in each node of the stage. The number of edges entering each stage is $2|\mathcal{S}|$ because there are exactly two transitions starting from each node of the previous stage. Therefore, the number of operations at each stage of G, is proportional to $|\mathcal{S}|$. Thus, the total time requirement equals $O(M|\mathcal{S}|) = O(MK^2 + MN \log N)$, or, equivalently, $O(MK^2 + MN \log N)$ since $T = \beta M$.

Let us now analyze the space complexity. Seemingly, for each node of G, the last edge of the maximum-weight path ending in that node needs to be stored, as required by backtracking of a typical dynamic programming process. This would imply an $O(M|\mathcal{S}|) = O(MK^2 + MN \log N)$ space complexity. However, by a careful examination of the trellis one can see that only knowing the complete nodes on a path is sufficient for reconstructing the whole path because the intermediate nodes can be generated by stepping through the (deterministic) convolutional encoder. Therefore, for each surviving path ending in a complete node, the last visited complete node is stored. Then the space necessary for joint source-channel MAP decoding becomes $O(M|\mathcal{S}'|)$, where $\mathcal{S}'$ is the set of complete states of the RCFSM. The evaluation of $|\mathcal{S}'|$ follows the same line of thought as in Section IV. For a triple (c_i, L_1, S_k) of the CFSM to be reachable from the initial state, either c_i has to be a suffix of S_k or vice versa. If $|c_i| \geq m$, its m-bit suffix

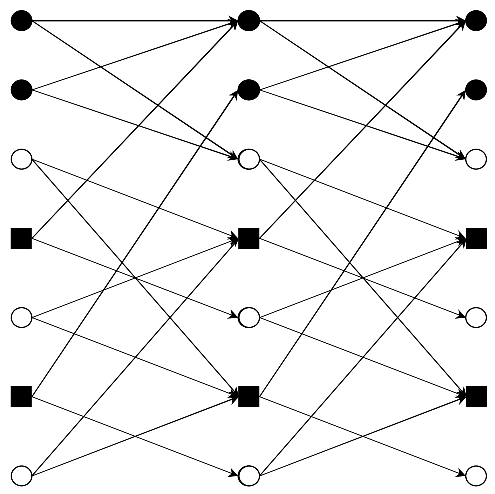

Fig. 4. Decoder graph G: three consecutive stages and edges between them; black circles and squares are nodes corresponding to the completion of a source codeword; the squares correspond to codewords of at least m bits, and the circles to the rest.

corresponds to exactly one state S_k of the CC. If $|c_i| < m$, then there are $2^{m-|c_i|}$ CC states S_k which have c_i as a suffix. Using further Kraft's inequality we obtain that

$$\sum_{i,|c_i|<m} 2^{m-|c_i|} = N \sum_{i,|c_i|<m} 2^{-|c_i|} \leq N. \tag{10}$$

It follows that $|\mathcal{S}'| \leq K + N$. Thus, the space complexity becomes $O(MK + MN) = O(TK + TN)$.

Before ending this section we briefly address the case of rate α/β finite input response CC with $\alpha > 1$. In this case each state of the CC has to be split into α states. Thus, the CC can be described by a state diagram with one input bit transitions. By the same arguments as for a rate $1/\beta$ CC, we obtain the reduction of the CFSM's state space size to $O(\alpha K^2 + N \log N)$. This leads to an MAP decoding algorithm of $O(\alpha TK^2 + TN \log N)$ time complexity, which represents a drastic improvement over the complexity of $O(TK^2N)$ as known currently. A similar observation as in the case of $(\beta, 1)$ CC leads to a reduction of the space requirement from $O(\alpha TK^2 + TN \log N)$ to $O(\alpha TK + TN)$.

VI. Alternative Decoder Graph

We will show in Section 7 how the MAP decoding time can be decreased if the logarithm of the Markov source transition probabilities satisfy the so-called Monge property, which is the main contribution of this work. For this we need first to construct an alternative decoder graph G' from G. The general idea in obtaining the graph G' from G is that any sequences of edges which corresponds to decoding a codeword of length at least m (the memory of the CC), and which starts from some complete node corresponding to a codeword of length at least m, is "contracted" into a single edge. By this contraction we do not understand that all the edges in the original sequence are removed, but only those which are not shared with other non-contracted sequences. This concept will be made clear in the following paragraphs.

Let $\mathcal{N}_c$ denote the set of complete nodes in the graph G which correspond to decoding a source codeword of at least m bits. As we have seen in Section IV, for each i such that $|c_i| \geq m$, there is a unique k such that (c_i, L_1, S_k) is in the RCFSM, i.e., S_k has to be the m-bit suffix of c_i. We denote this value of k by $k(i)$. Thus, $\mathcal{N}_c$ is the set of all nodes $(c_i, L_1, S_{k(i)})_t$ in the graph G such that $|c_i| \geq m$ and $t \geq 0$.

The construction of G' from G is as follows. If two nodes in $\mathcal{N}_c$ are connected by a path which does not visit other complete nodes, then this path is contracted into a single edge. The original path is not necessarily completely removed, but only those edges which are not shared with other paths connecting consecutive complete nodes at least one of which is not in $\mathcal{N}_c$. To explain this process better let us consider such a path which is contracted into a single edge. Let the starting node of the path be $(c_i, L_1, S_{k(i)})_t$ for some $1 \leq i \leq K$ such that $|c_i| \geq m$, and $t \geq 0$. Then the ending node must be $(c_{i'}, L_1, S_{k(i')})_{t+|c_{i'}|}$ for some $1 \leq i' \leq K$ such that $|c_{i'}| \geq m$. Any intermediate node on this path is of the form $(c_i, L_j, S_k)_{t+|L_j|}$ where $L_j \neq L_1$, L_j is an ascendant of $c_{i'}$ in the source code tree, and S_k is the m-bit suffix of $c_i L_j$. Such a graph node is also contained in any path connecting $(c_i, L_1, S_{k(i)})_t$ with some node $(c_{i"}, L_1, S_{k(i")})_{t+|c_{i"}|}$ provided that L_j is an ascendant of $c_{i"}$ in the source code tree. If $|c_{i"}| < m$, then the path from $(c_i, L_1, S_{k(i)})_t$ to $(c_{i"}, L_1, S_{k(i")})_{t+|c_{i"}|}$ is not contracted, consequently the node $(c_i, L_j, S_k)_{t+|L_j|}$ cannot be removed. On the other hand, if each codeword descending from L_j in the source code tree has at least m bits, then the node $(c_i, L_j, S_k)_{t+|L_j|}$ can be removed together with its incoming and outgoing edges. We illustrate the difference between the two graphs G and G' on the example considered in the previous sections. Figures 4 and 5 show the edges between three consecutive stages in the two graphs respectively.

In conclusion, the new graph G' is obtained from G by removing all nodes $(c_i, L_j, S_k)_t$ such that $t \geq 0$, $|c_i| \geq m$, and L_j is not an ascendant of any codeword $c_\imath$ with $|c_\imath| < m$ in the source

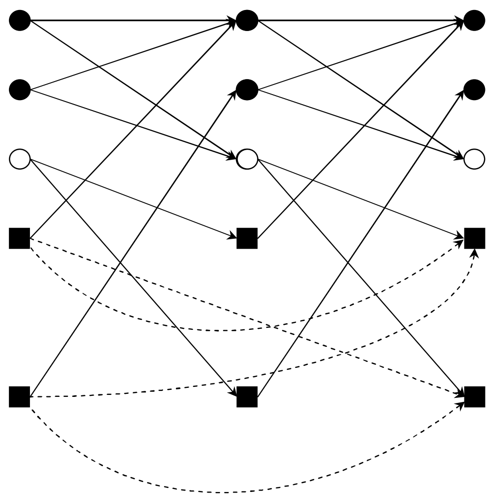

Fig. 5. New decoder graph G': three consecutive stages and edges between them (only edges with both ends at these stages are illustrated); black circles and squares are nodes corresponding to the completion of a source codeword; the squares correspond to codewords of at least m bits, and the circles to the rest.

code tree. All edges entering or leaving such nodes are removed from G too. Further, we add an edge from $(c_i, L_1, S_{k(i)})_t$ to $(c_{i'}, L_1, S_{k(i')})_{t+|c_{i'}|}$, for any $1 \leq i, i' \leq K$ such that $|c_i| \geq m$, $|c_{i'}| \geq m$, and $t \geq 0$, and set its weight to be equal to the weight of the old path between the two nodes (note that the old path between the two nodes was unique).

Clearly, the MAP sequence decoding problem is equivalent to finding the maximum-weight path in G'. This could be done by advancing sequentially through the stages of the graph G' and computing at each stage the maximum-weight path for each node at that stage. However, in order to reduce the complexity, we organize the computations differently. Namely, we still proceed sequentially through the stages of the graph G', but at each stage t, we find the maximum-weight path for each node at stage $t+1$ andadditionally we process the paths ending at nodes of later stages ($t + |c_{i'}|$ with $|c_{i'}| \geq m$), but whose last visited node is at stage t.

In order to give a precise description of the algorithm we need a few more notations. For each node n of the new graph G', let $W(n)$ denote the weight of the maximum-weight path from the source to n. For the nodes $n \in \mathcal{N}_c$ we compute $W(n)$ by considering separately the paths whose last visited node is in $\mathcal{N}_c$, and the rest of the paths. Let $W_1(n)$ and $W_2(n)$ denote the weight of the maximum-weight path in the two situations respectively. Then clearly $W(n)$ is the maximum of $W_1(n)$ and $W_2(n)$. For any node $n' \in \mathcal{N}_c$ situated at stage $t + |c_{i'}|$ for some

t and i' with $|c_{i'}| \geq m$, the value $W_1(n')$ is computed at stage t because all paths ending in n' with the last visited node in $\mathcal{N}_c$, have the last visited node situated at stage t. Further, at stage $t + |c_{i'}| - 1$, the value $W_2(n')$ is computed too and compared against $W_1(n')$ in order to get $W(n)$.

We denote by $\mathcal{N}_c(t)$ (respectively, $\bar{\mathcal{N}_c}(t)$) the set of nodes of G' at stage t which are (respectively, are not) in $\mathcal{N}_c$. Denote an edge from node n to node n' by (n, n') and its weight by $w(n, n')$. Let E' be the set of all edges of G'.

At each stage t the algorithm performs the following computations:

a) Compute $W(n')$, for all $n' \in \bar{\mathcal{N}_c}(t+1)$, according to

$$W(n') = \max_{n \in \mathcal{N}_c(t) \cup \bar{\mathcal{N}_c}(t), (n,n') \in E'} \{W(n) + w(n, n')\}. \tag{11}$$

b) Compute $W_2(n')$, for all $n' \in \mathcal{N}_c(t+1)$, according to

$$W_2(n') = \max_{n \in \bar{\mathcal{N}_c}(t), (n,n') \in E'} \{W(n) + w(n, n')\}. \tag{12}$$

c) Compute $W_1(n')$ for all $n' \in \mathcal{N}_c(t + |c_{i'}|)$ (i.e., for all $n' = (c_{i'}, L_1, S_{k(i')})_{t+|c_{i'}|}$ for some i' such that $|c_{i'}| \geq m$) using

$$W_1(n') = \max_{n \in \mathcal{N}_c(t)} \{W(n) + w(n, n')\}. \tag{13}$$

d) Compute $W(n')$, for all $n' \in \mathcal{N}_c(t+1)$, according to

$$W(n') = \max\{W_1(n'), W_2(n')\}. \tag{14}$$

Note that relation (14) can be applied because the values $W_1(n')$ have been computed at previous stages.

VII. Accelerated Decoding by Fast Matrix Search

We show next how the algorithm outlined in the previous section can be sped up if the logarithm of the Markov source transition probabilities satisfy the Monge property, i.e.:

$$\log P(a_{i'_1}|a_{i_1}) + \log P(a_{i'_2}|a_{i_2}) \geq$$
$$\log P(a_{i'_1}|a_{i_2}) + \log P(a_{i'_2}|a_{i_1})$$
$$\text{for all } 1 \leq i_1 < i_2 < K \text{ and } 1 \leq i'_1 < i'_2 \leq K, \tag{15}$$

It was shown in [20] that if the discrete Markov source is the output of a scalar quantizer applied to a continuous source[1] whose joint pdf $f(u,v)$ of two consecutive samples satisfies

$$\log f(v',u) + \log f(v,u') \leq \log f(v',u') + \log f(v,u), \tag{16}$$

for any real values $u < u'$ and $v < v'$, then the condition (15) holds. If the second partial derivative $\partial^2(\log f)/\partial u \partial v$ exists, then (16) holds if and only if $\partial^2(\log f)/\partial u \partial v \geq 0$ for all real u, v [3]. This is clearly true when the joint pdf $f(\cdot,\cdot)$ is Gaussian. Consequently, the Monge property (15) holds for a scalar quantized Gaussian-Markov source.

To speed up the MAP decoding algorithm we need to carefully organize the computations in task c). Instead of solving the instances of the maximization problem separately, we structure these instances into a matrix as follows. Let us assume without restricting the generality, that the source codewords c_i such that $|c_i| \geq m$, are $c_1, c_2, \cdots, c_{K_c}$, for some $K_c \leq K$. Further consider the $K_c \times K_c$ matrix $\mathcal{M}_t$ with elements $\mathcal{M}_t(i,i')$, $1 \leq i, i' \leq K_c$, defined as follows:

$$\mathcal{M}_t(i,i') = W(n) + w(n,n'), \tag{17}$$

where $n = (c_i, L_1, S_{k(i)})_t$ and $n' = (c_{i'}, L_1, S_{k(i')})_{t+|c_{i'}|}$. Then (13) becomes

$$W_1((c_{i'}, L_1, S_{k(i')})_{t+|c_{i'}|}) = \max_{i, 1 \leq i \leq K_c} \mathcal{M}_t(i,i'). \tag{18}$$

In other words, computing $W_1((c_{i'}, L_1, S_{k(i')})_{t+|c_{i'}|})$ is equivalent to finding the maximum element of column i' of the matrix $\mathcal{M}_t$. Hence task c) is transformed to a matrix search problem on $\mathcal{M}_t$. If the matrix satisfies a property known as total monotonicity, then this problem can be solved more efficiently [2].

A matrix $\mathcal{M}$ is said to be totally monotone [2] if

$$\begin{gathered}\mathcal{M}(i_1,i'_1) \leq \mathcal{M}(i_2,i'_1) \Longrightarrow \mathcal{M}(i_1,i'_2) \leq \mathcal{M}(i_2,i'_2) \\ \text{for all } i_1 < i_2 \text{ and } i'_1 < i'_2.\end{gathered} \tag{19}$$

Our matrix $\mathcal{M}_t$ is not totally monotone, in general, but it can be divided into sub-matrices for which the condition (19) holds if (15) holds. For this, the rows of matrix $\mathcal{M}_t$ are partitioned into subsets such that for all rows i in a given subset, $k(i)$ is the same (i.e., all codewords c_i have the same m-bit suffix).

[1]Although a quantized Markov chain may not necessarily be a Markov chain, here we assume this to be a good approximation.

We may assume without loss of generality that the indexing of the codewords $c_1, c_2, \cdots, c_{K_c}$ is such that the codewords with common m-bit suffix have consecutive indeces. Then there are integers $L, 1 \leq L \leq N$, and $1 = m_0 < m_1 < \cdots < m_L = K_c + 1$, such that for any $l, 1 \leq l \leq L$, $k(i)$ is the same for all $i, m_{l-1} \leq i < m_l$. For each $l, 1 \leq l \leq L$, we define the matrix $\mathcal{M}_{t,l}$ as the sub-matrix of $\mathcal{M}_t$ formed with the rows $m_{l-1}, m_{l-1}+1, \cdots, m_l - 1$.

Proposition. If condition (15) holds, then all sub-matrices $\mathcal{M}_{t,l}$ are totally monotone.

Proof. An obvious sufficient condition for the sub-matrix $\mathcal{M}_{t,l}$ to be totally monotone is

$$\begin{gathered}\mathcal{M}_t(i_1, i'_1) + \mathcal{M}_t(i_2, i'_2) \geq \mathcal{M}_t(i_2, i'_1) + \mathcal{M}_t(i_1, i'_2) \\ \text{for all } m_{l-1} \leq i_1 < i_2 < m_l \text{ and } 1 \leq i'_1 < i'_2 \leq K_c.\end{gathered} \tag{20}$$

Recall that $\mathcal{M}_t(i, i') = W(n) + w(n, n')$, where $n = (c_i, L_1, S_{k(i)})_t$ and $n' = (c_{i'}, L_1, S_{k(i')})_{t+|c_{i'}|}$. The value $w(n, n')$ is the weight of the edge from n to n' of the graph G', which was defined as the weight of the path from n to n' in the graph G. This path has exactly $|c_{i'}|$ edges, and it corresponds to a sequence of $|c_{i'}|$ transitions in the FSM of MS$\rightarrow$SC from the state (c_i, L_1) to the state $(c_{i'}, L_1)$, whose output is the bitstream forming the codeword $c_{i'}$. Also this path corresponds to the sequence of transitions of the CC starting from the state $S_{k(i)}$, and generated by the the input $c_{i'}$. Let $b_1 b_2 \cdots b_{\beta|c_{i'}|}$ be the output of this sequence of transitions. Then, the weight of this path, hence the weight of the edge (n, n') of G', is $w(n, n') = \log P_e(y_{t\beta+1} \cdots y_{(t+|c_{i'}|)\beta} | b_1 b_2 \cdots b_{\beta|c_{i'}|}) +$ $\log P(a_{i'}|a_i)$. Note that the first term in the above equality depends only on three parameters, namely t, $k(i)$, and i'. Therefore, we will simply denote it by $\omega(t, k(i), i')$. Then the entry $\mathcal{M}_t(i, i')$ can be written as

$$\mathcal{M}_t(i, i') = W(n) + \omega(t, k(i), i') + \log P(a_{i'}|a_i). \tag{21}$$

By substituting (21) in (20) and making all cancellations (note that since $m_{l-1} \leq i_1 < i_2 < m_l$, we have $k(i_1) = k(i_2)$) we obtain that (20) is equivalent to

$$\begin{gathered}\log P(a_{i'_1}|a_{i_1}) + \log P(a_{i'_2}|a_{i_2}) \geq \\ \log P(a_{i'_1}|a_{i_2}) + \log P(a_{i'_2}|a_{i_1}) \\ \text{for all } m_{l-1} \leq i_1 < i_2 < m_l \text{ and } 1 \leq i'_1 < i'_2 \leq K_c,\end{gathered} \tag{22}$$

which clearly holds when (15) holds. $\square$

The fast matrix search technique proposed by Aggarwal *et al.* [2] solves the matrix search problem in $O(r_1(1+\log(r_2/r_1)))$ time, if the matrix has r_1 rows and r_2 columns ($r_1 \leq r_2$), and it is totally monotone. Consequently, if all matrices $\mathcal{M}_{t,l}$, $1 \leq l \leq L$, are totally monotone, solving the matrix search for all of them can be done in $O(\sum_{l=1}^{L}(m_l-m_{l-1})(1+\log(K_c/(m_l-m_{l-1})))) = O(K_c \log K_c)$ time. Further, in order to find the maximum element on each column of $\mathcal{M}_t$, for each column the maximum over the maxima in the L sub-matrices $\mathcal{M}_{t,l}$ has to be computed, taking additional $O(K_c L)$ time. Thus, according to Proposition, if the source satisfies condition (15), then the time required to perform the task c) is $O(K_c(\log K_c + L))$.

The total number of operations necessary to perform tasks a) and b) and d) is proportional to $|\mathcal{N}_c(t)| + |\bar{\mathcal{N}_c}(t)|$. Clearly, $|\mathcal{N}_c(t)| = K_c$. To evaluate the number of nodes in $\bar{\mathcal{N}_c}(t)$, note that such nodes are all quadruples $(c_i, L_j, S_k)_t$ such that (c_i, L_j, S_k) is in the RCFSM, and one of the following conditions is satisfied:

i) $|c_i| < m$;

ii) $|c_i| \geq m$, and L_j is a ascendant of some codeword $c_{i'}$ with $|c_{i'}| < m$, in the source code tree.

By the same line of argument as in Section IV, we conclude that the number of nodes satisfying condition i) is at most $(K - K_c)K + N \log_2 N$, while the number of nodes satisfying condition ii) is at most $K_c(K - K_c)$. Consequently, the total number of operations to perform tasks a), b) and d) is $O(K(K - K_c) + N \log_2 N)$. Adding this cost to the time requirements for task c), and multiplying the total by the number of stages M (which is proportional to T), we obtain the time complexity of the algorithm outlined in the previous section:

$$O(T(K(K - K_c) + N \log N + K_c(L + \log K_c))). \tag{23}$$

Clearly, the improvement in efficiency is greater as $K - K_c$ and L become smaller, which is the case when the source code tree is more unbalanced. For fixed-length source code with $K > N$, we have $K_c = K$ and $L = N$. Thus, the decoding time using the fast matrix search technique is $O(T(N \log N + KN))$. For the Golomb-Rice code with $K > \log_2 N$ we have $K_c = K - \log_2 N + 1$ and $L = 2$, thus the decoding using the fast matrix search takes $O(T(K \log K + K \log N + N \log N))$ time.

It is easy to see that the fast matrix search approach can still be applied if the relation (15) holds, after a relabeling of the codewords, or more generally, if there are two permutations ϕ

and ψ on the integers between 1 and K_c, such that

$$\log P(c_{\phi(i)}|c_{\psi(j')}) + \log P(c_{\phi(i')}|c_{\psi(j)}) \leq$$
$$\log P(c_{\phi(i')}|c_{\psi(j')}) + \log P(c_{\phi(i)}|c_{\psi(j)})$$
$$i < i', j < j'. \tag{24}$$

In this case the rows and columns of each matrix $\mathcal{M}_{t,l}$ have to be permuted by using ϕ, respectively ψ. An $O(K^2)$ time algorithm to check if such permutations exist can be found in [5], [3]. The decoder can first apply this algorithm in order to decide whether to use the fast matrix search technique or not. The extra cost required by this test does not alter the asymptotic complexity of the chosen algorithm.

VIII. Conclusion

This paper reexamines the complexity of joint source-channel decoding of a Markov sequence which is first encoded by a source code, then by a finite impulse response convolutional encoder, and sent through a noisy memoryless channel. It was established previously that the joint source-channel decoding can be performed by using a Viterbi-like algorithm on a bit-level trellis whose state space size at each level is $O(K^2N)$, where K is the number of Markov symbols, and N is the number of convolutional encoder states. We show that the state space size can be drastically reduced to $O(K^2 + N \log N)$. A further reduction can be achieved if the logarithm of the source transition probabilities satisfies the Monge property. The reduction factor ranges from $O(K/N)$ (for a fixed-length source code) to $O(K/\log N)$ (for Golomb-Rice code), increasing as the tree structure of the source code becomes more unbalanced.

IX. Appendix

In this appendix we present the derivations of the precise count of the significant states of the RCFSM in the case of Golomb-Rice code and fixed-length source codes.

For the Golomb-Rice code, each codeword c_i, for $1 \leq i \leq K-1$, is the bit sequence of $i-1$ 1's followed by a 0, and c_K is the bit sequence of $K-1$ 1's. For each $j, 2 \leq j \leq K-1$, L_j is the bit sequence of $j-1$ 1's. We assume that $m \leq K-1$ As we have seen $|\mathcal{C}_I|$ equals the number of ordered pairs (i, j), $1 \leq i \leq K$, $1 \leq j \leq K-1$, which satisfy the inequality $m \leq |c_i| + |L_j|$. Consequently,

$$|\mathcal{C}_I| = \sum_{i=1}^{K} \sum_{j, |L_j| \geq m - |c_i|} 1. \tag{25}$$

Since $|c_i| = i$ for $i \leq K-1$, $|c_K| = K-1$, and $|L_j| = j-1$ for all j, we further obtain

$$\begin{aligned}
|\mathcal{C}_I| &= \sum_{i=1}^{K-1} \sum_{j, j \geq m-i+1} 1 + \sum_{j, j \geq m-K+2} 1 \\
&= \sum_{i=1}^{m} \sum_{j=m-i+1}^{K-1} 1 + \sum_{i=m+1}^{K-1} \sum_{j=1}^{K-1} 1 + \sum_{j=1}^{K-1} 1 \\
&= \sum_{i=1}^{m} (K-1-m+i) + \sum_{i=m+1}^{K-1} (K-1) + K - 1 \\
&= m(K-1-m) + \sum_{i=1}^{m} i + (K-1)(K-m) \\
&= K^2 - K - \frac{m^2 - m}{2}.
\end{aligned} \tag{26}$$

We mention that the second equality in the series of relations above was obtained by breaking in two parts the summation over i and by using the observation that when $i > m$, the inequality $j \geq m - i + 1$ is satisfied for all $j, 1 \leq j \leq K-1$. Further, according to (3) we have

$$|\mathcal{C}_{II}| = 2^m \sum_{i,j, |c_i| + |L_j| < m} 2^{-|c_i| - |L_j|}. \tag{27}$$

Note that $|c_i| + |L_j| < m$ implies $i \leq m-1$, consequently,

$$\begin{aligned}
|\mathcal{C}_{II}| &= 2^m \sum_{i,j, |c_i| + |L_j| < m} 2^{-|c_i| - |L_j|} \\
&= 2^m \sum_{i=1}^{m-1} 2^{-i} \sum_{j=1}^{m-i} 2^{-(j-1)} \\
&= 2^m \sum_{i=1}^{m-1} 2^{-i} (2 - 2^{-(m-i-1)}) \\
&= 2^{m+1} - 2m - 2.
\end{aligned} \tag{28}$$

Relations (26) and (28) imply that

$$|\mathcal{S}| = K^2 - K + 2N - \tfrac{1}{2}(\log_2^2 N + 3 \log_2 N + 4). \tag{29}$$

Consider now a fixed-length source code with K being an integer power of 2. Note that the length of each codeword is $\log_2 K$. We need to distinguish between three cases.

i) $N \leq K$. In this case $\mathcal{C}_{II}$ is empty and $\mathcal{C}_I$ contains a state for each pair c_i, L_j. Then

$$|\mathcal{S}| = K^2 - K. \tag{30}$$

ii) $K < N \leq K^2/2$. In this case we have

$$\begin{aligned} |\mathcal{C}_I| &= \sum_{i,j,|c_i|+|L_j|\geq m} 1 \\ &= \sum_{i,j,|L_j|\geq m-\log_2 K} 1 \\ &= K \sum_{l=m-\log_2 K}^{\log_2 K-1} \sum_{j,|L_j|=l} 1 \\ &= K \sum_{l=m-\log_2 K}^{\log_2 K-1} 2^l \\ &= K(2^{\log_2 K} - 2^{m-\log_2 K}) \\ &= K^2 - N. \end{aligned} \tag{31}$$

Further, according to (3) we obtain

$$\begin{aligned} |\mathcal{C}_{II}| &= 2^m \sum_{i,j,|c_i|+|L_j|<m} 2^{-|c_i|-|L_j|} \\ &= N \sum_{i,j,|L_j|<m-\log_2 K} 2^{-\log_2 K-|L_j|} \\ &= NK2^{-\log_2 K} \sum_{l=0}^{m-\log_2 K-1} \sum_{j,|L_j|=l} 2^{-l} \\ &= N \sum_{l=0}^{m-\log_2 K-1} 2^{-l}2^l \\ &= N(\log_2 N - \log_2 K). \end{aligned} \tag{32}$$

It follows that

$$|\mathcal{S}| = K^2 + N\log_2 N - N(1+\log_2 K). \tag{33}$$

iii) $N > K^2/2$. In this case $\mathcal{C}_I$ is empty and

$$|\mathcal{C}_{II}| = N \sum_{l=0}^{m-\log_2 K-1} \sum_{j,|L_j|=l} 2^{-l} =$$
$$N \sum_{l=0}^{\log_2 K-1} 2^{-l} 2^l = N \log_2 K. \tag{34}$$

Then

$$|\mathcal{S}| = N \log_2 K. \tag{35}$$

References

[1] F. Alajaji, N. Phamdo, N. Farvardin and T. Fuja, ”Detection of binary Markov sources over channels with additive Markov noise”, *IEEE Trans. Inform. Th.*, vol. 42, no. 1, pp. 230-239, Jan. 1996.

[2] A. Aggarval, M. Klave, S. Moran, P. Shor, and R. Wilber, ” Geometric applications of a matrix-searching algorithm”, *Algorithmica*, 2(1987), pp.195-208.

[3] R. E. Burkard, B. Klinz, and R. Rudolf, Perspectives of Monge properties in optimization, *Discrete Applied Mathematics* 70, 1996, pp. 95-161.

[4] Q. Chen and K. P. Subbalakshmi, ”An Integrated Joint Source-Channel Decoder for MPEG-4 Coded Video”, *Proc. IEEE Vehicular Tech. Conf. 2003*, vol. 1, pp. 347-351.

[5] V. G. Deineko and V. L. Filonenko, ”On the reconstruction of specially structured matrices”, *Aktualnyje Problemy EVM i programmirovanije*, Dnepropetrovsk, DGU, 1979, (in Russian).

[6] J. Garcia-Frias and J. D. Villasenor, ”Combining Hidden Markov Source Models and Parallel Concatenated Codes”, *IEEE Comm. Letters*, vol. 1, no. 4, pp. 111-113, July 1997.

[7] A. Guyader, E. Fabre, C. Guillemot, and M. Robert, ”Joint source-channel turbo decoding of entropy-coded sources”, *IEEE Journal on Selected Areas in Comm.*, vol. 19, no. 9, pp. 1680-1696, Sept. 2001.

[8] J. Hagenauer and N. Gortz, ”The Turbo Principle in Joint Source-Channel Coding”, *IEEE ITW 2003*, pp. 275-278, April 2003.

[9] S. Lin, D. J. Costello Jr., *Error Control Coding*, Pearson Prentice Hall, New Jersey, 2004.

[10] A. H. Murad and T. E. Fuja, ”Joint source-channel decoding of variable-length encoded sources”, *IEEE ITW’98*, June 1998, pp. 94-95.

[11] N. Phamdo and N. Farvardin, ”Optimal Detection of Discrete Markov Sources over Discrete Memoryless Channels - Applications to Combined Source-Channel Coding”, *IEEE Trans. Inf. Th.*, vol. 40, no.1, pp. 186-193, Jan. 1994.

[12] N. Phamdo, F. Alajaji, and N. Farvardin, ”Quantization of memoryless and Gauss-Markov sources over binary Markov channels”, *IEEE Trans. Comm.*, vol. 45, no. 6, pp. 668-675, June 1997.

[13] M. Park and D. J. Miller, “Joint source-channel decoding for variable-length encoded data by exact and approximated MAP sequence estimation”, *IEEE Trans. Comm.*, vol. 48, no. 1, pp. 1-6, Jan. 2000.

[14] K. Sayood and J. C. Borkenhagen, ”Use of Residual Redundancy in the Design of Joint Source/Channel Coders”, *IEEE Trans. Comm.*, vol. 39, no. 6, pp. 835-846, June 1991.

[15] M. Skoglund and P. Hedelin, ”Hadamard-based soft decoding for vector quantization over noisy channels”, *IEEE Trans. Inform. Th.*, vol. 45, no. 2, pp. 515-532, March 1999.

[16] M. Skoglund, ”Soft decoding for vector quantization over noisy channels with memory”, *IEEE Trans. Inform. Th.*, vol. 45, no. 4, pp. 1293-1307, May 1999.

[17] K. P. Subbalakshmi and J. Vaisey, “On the Joint Source-Channel Decoding of Variable-Length Encoded Sources: The BSC Case”, *IEEE Trans. Comm.*, vol. 49, no. 12, pp. 2052-2055, Dec. 2001.

[18] M. Skoglund and P. Hedelin, ”Hadamard-based soft decoding for vector quantization over noisy channels”, *IEEE Trans. Inform. Th.*, vol. 45, no. 2, pp. 515-532, March 1999.

[19] M. Skoglund, ”Soft decoding for vector quantization over noisy channels with memory”, *IEEE Trans. Inform. Th.*, vol. 45, no. 4, pp. 1293-1307, May 1999.

[20] X. Wu, S. Dumitrescu, and Z. Wang, ”Monotonicity-based fast algorithm for MAP estimation of Markov sequences over noisy channels”, *IEEE Trans. Inf. Theory*, vol. 50, pp. 1539-1544, July 2004.