JOINT SOURCE-CHANNEL TURBO TECHNIQUES FOR WIRELESS MULTIMEDIA COMMUNICATION

Christine Guillemot

IRISA

Campus universitaire de Beaulieu
35042 Rennes Cédex
Christine.Guillemot@irisa.fr

Outline

- Applications and problems addressed
- Soft-decision source decoding
- Source-channel turbo coding/decoding
- Source codes with properties for source-channel coding
- Source codes with inherent resilience
- Conclusion

Outline

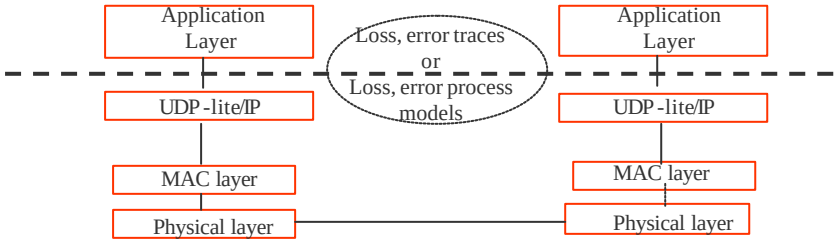
- Applications and problems addressed
- Soft-decision source decoding
- Source-channel turbo coding/decoding
- Source codes with properties for source-channel coding
- Source codes with inherent resilience
- Conclusion

Application Domains (1)

- Communication over networks supporting transparent modes, e.g.:
  - In the MAC layer: 3GPP, Bluetooth, …
  - In the transport layer: UDP-lite, DCCP
- Digital media storage (DVD, CD):
  - Trend to relax the drastic residual error rate (10**-20 => up to 25% redundancy
  - Reducing the need for error correcting codes (hence increasing the storage capacity) while limiting effects of errors on quality
- Transmissions not relying on IP, e.g.:
  - Broadcasting applications such as Digital Radio Mondiale

Application domains (2): Example

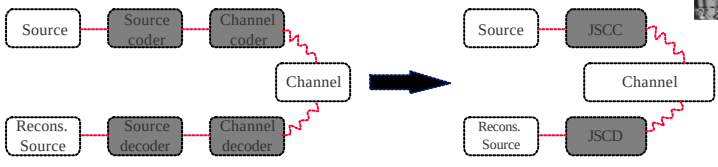
- Communication with transparent modes to save bandwidth



	BSC		3GPP		WIFI	
	UDP	UDP-Lite	UDP	UDP-Lite	UDP	UDP-Lite
mean number of received packets	6422	9466	9751	9887	9112	9742
% mean gain	47,4		1,4		6,9	

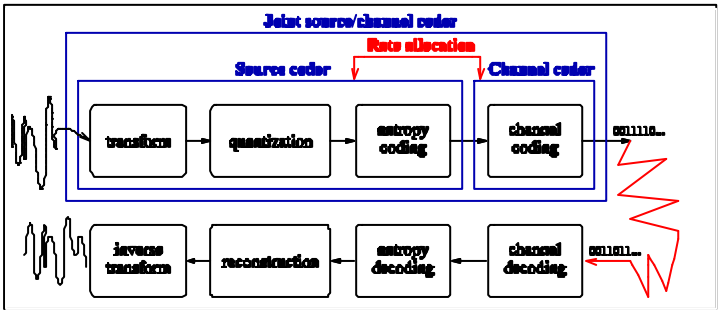
Introduction to the problem (1)

- Optimize the end-to-end QoS on wireless links
  - Narrow bands
  - Transmission noise (errors, erasures)
- In the traditional set-up
  - Source coding does not care about channel errors
    - High sensitivity of VLCs to errors
    - Decoder de-synchronisation problem
  - Use of error correcting codes



Introduction to the problem (2)

- Channel coding (ECC) is efficient against noise
  - But channel characteristics must be known at the time of encoding and may vary in time => residual bit errors
  - The separation theorem does not account for practical constraints (bounded delay, complexity)
- Unequal error protection (UEP)



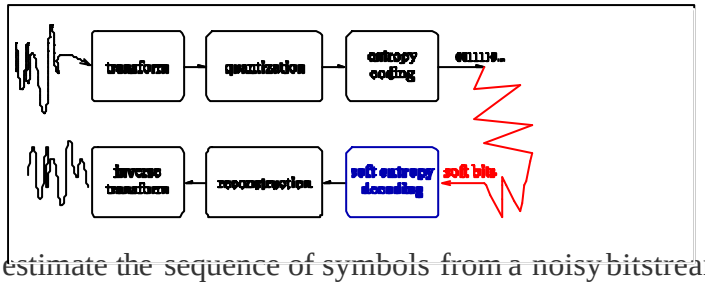
Introduction to the problem (3)

- Self-synchronization of VLCs has been extensively studied, e.g. [Maxted & al. 85, Zhou & al. 02]
  - "self-synchronization": in terms of Levenshtein distance (not strict sense synchronization)
- Design of Reversible Variable Length Codes (RVLCs) [Takishima & al. 95]
- Robust decoding methods exploiting the sub-optimality of the source code
  - Bayesian Estimation => Soft-Decision Source Decoding [Balakirsky 97]
  - Turbo-VLC [Hagenauer00, Guyader, Fabre & Guillemot 01, …]

Outline

- Applications and problems addressed
- Soft decoding of VLCs
  - Tree codes (e.g., Huffman)
  - Soft synchronisation
  - Arithmetic codes
  - Trellis and complexity issues
  - Application to JPEG-2000 and to H.264/MPEG-4 AVC
- Source-channel turbo coding/decoding
- Source codes with properties for source-channel coding
- Source codes with inherent resilience features
- Conclusion

Soft Decoding of VLCs

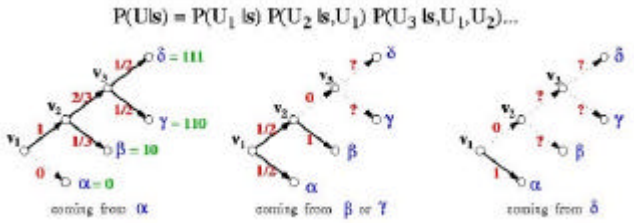


Goal: - To estimate the sequence of symbols from a noisy bitstream by exploiting the correlation in the chainand the sub-optimality of the code

=> Modelling the dependencies between the processes of the chain with a Bayesian network formalism

Soft Decoding of VLCs

- Modelling the VLC process



- Bit clock model obtained by concatenation of codetrees or merging the leaves with the root node

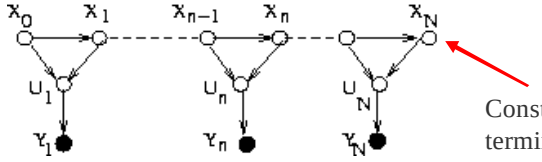
$$X_n = (v_n)$$

Vertex of codetree (internal node)

- The sequence of states X_n forms a Markov chain

Soft Decoding of VLCs

- Bit clock state model and estimator (memoryless source)



Constraint to terminate at the root node

=> Bit-level trellis of Balakirsky [Balakirsky 97]

- MAP estimator

$$\widehat{X}_1 \ldots \widehat{X}_n = \arg\max_{X_1 \ldots X_n} P(X_1 \ldots X_n / Y_1, Y_2, \cdots Y_N)$$

=> SOVA algorithm [Hagenauer & Hoeher 89]

- MPM estimator

$$\widehat{X}_n = \arg\max_{X_n} P(X_n / Y_1, Y_2, \cdots Y_N)$$

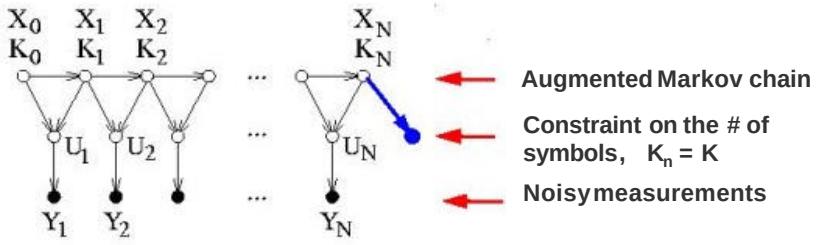
$$P(X_n / Y_1, Y_2, \cdots Y_N) \alpha\ P(X_n / \overline{Y}_1^n) \times P(\overline{Y}_{n+1}^N / X_n)$$

=> Two recursions BCJR algorithm [Bahl, Cocke, Jelinek, Raviv, 97]

Soft Decoding of VLCs

- To force an additional termination constraint

$(X_n, K_n) = (v_n, k)$



=> Bit/symbol trellis [Hagenauer 00]

- Optimum performance but complexity growing as a quadratic function of the sequence length

Soft Decoding of VLCs

- Accounting for symbol correlation (source with memory)
  - Markov source + VLC coding process: Product Model

$(X_n, K_n) = (s,v, k)$

The model keeps track of the last symbol produced

=> Augmented bit/symbol trellis

- Tree-structure graph amenable to fast estimation algorithms (BCJR, Viterbi)
- But state space of high dimension (untractable in practice)

- One can instead proceed in 2 steps [Guyader, Fabre & Guillemot 01]

Soft Decoding of VLCs

1- Bit clock : estimate $(X_n, K_n) = (v, k)$ assuming independant symbols, exploiting the inner codeword correlation

$\Rightarrow$ State space of reduced dimension

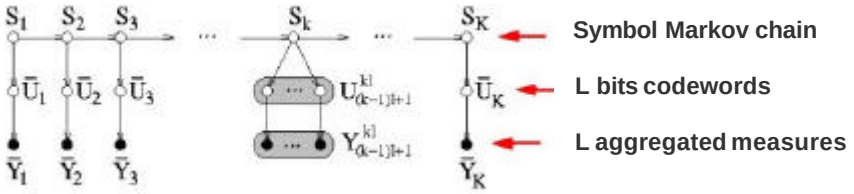
$\Rightarrow$ But sub-optimal (inter-symbol correlation not exploited)

2- To exploit the inter-symbol correlation via an estimation run on the Markov model of the sequence of symbols

$\Rightarrow$ Need for a symbol clock Markov model of the sequence of symbols

Soft Decoding of VLCs

- Symbol clock model: the simple case of FLCs:



- Fast estimation algorithms - MAP or MPM – on (S_k)

$$\hat{S}_k = \arg\max_{s_k} P(S_k/Y_1, Y_2, \cdots Y_N) = \arg\max_{s_k} P(S_k/\overline{Y}_1, \overline{Y}_2, \cdots \overline{Y}_K)$$

Two recursions algorithm(BCJR algorithm)

$$P(S_k/Y_1, Y_2, \cdots Y_N) \;\; a \;\; P(S_k/\overline{Y}_1^k) \times P(\overline{Y}_{k+1}^K / S_k)$$

$$P(S_k/\overline{Y}_1^k) \propto P(\overline{Y}_k/S_k) . \sum_{S_{k-1}} P(S_k/S_{k-1}) . P(S_{k-1}/\overline{Y}_1^{k-1})$$

$$P(\overline{Y}_k / S_k) = P(\overline{Y}_k / \overline{U}_k)$$

Local measurements ; Channel model

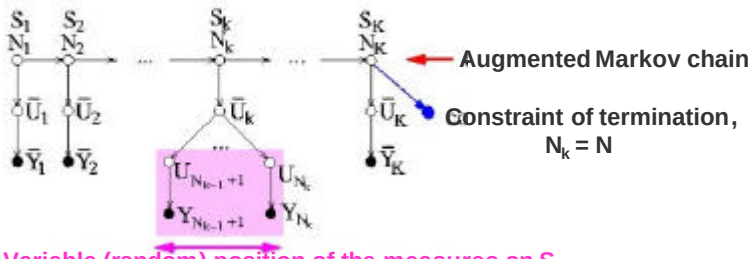
Soft Decoding of VLCs

- Symbol clock model for VLCs:
  - With VLCs, one has to locate S_k in the bitstream to compute the local measurements:

$$P(\overline{Y}_k/S_k,N_k) = P(\overline{Y}_{N_{k-1}+1}^{N_k}|\overline{U}_{N_{k-1}+1}^{N_k})$$

- Joint problem of segmentation and estimation

$$N_k = N_{k-1} + \mathcal{L}(S_k)$$



Soft Decoding of VLCs

1- Bit clock : estimate $(X_n, K_n) = (v, k)$ assuming independant symbols, exploiting the inner codeword correlation

⇒ State space of reduced dimension

⇒ But sub-optimal (inter-symbol correlation not exploited)

2- To transform the marginals on $(X_n, K_n) = (v, k)$ in marginals on (S_k, N_k) ⇒ Clock conversion on soft information, given the recursion $N_k = N_{k-1} + \text{length}(S_k)$

3- To incorporate the inter-symbol correlation

$$P(\overline{Y}_k/S_k,N_k) = P(\overline{Y}_{N_{k-1}+1}^{N_k}|\overline{U}_{N_{k-1}+1}^{N_k})$$

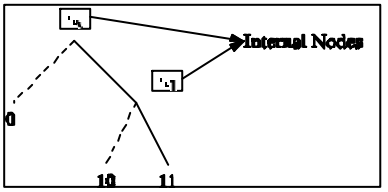
$$P(S_k/Y) \propto P(S_k/\overline{Y}_1^k).P(\overline{Y}_{k+1}^K|S_k)$$

$$P(S_k/\overline{Y}_1^k) \propto P(\overline{Y}_k/S_k).\sum_{S_{k-1}} P(S_k/S_{k-1}).P(S_{k-1}/\overline{Y}_1^{k-1})$$

This separate treatment of VLC coder and of Markov source is optimal

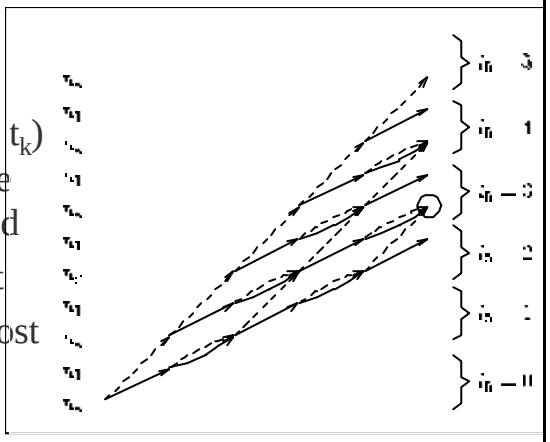
Trellis and complexity issues

- Bit-level trellis [Balakirsky 97]
  - The states are the internal nodes n_k
- Termination constraint
  - The trellis terminates in the root node
- Computing cost
  - O(|A|xL)
- VLC tables can be simplified
  - [Kieffer et al 05]



Trellis and complexity issues

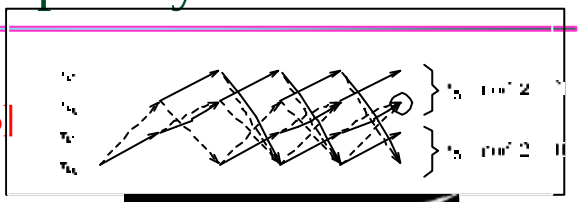
- Bit/symbol trellis [Park&Miller 98, Hagenauer00]
- States are defined as (n_k, t_k) so that the memory of the symbol clock is preserved
- Symbol length constraint
- Prohibitive computing cost
  - O(|A|xL^2)
  - Sub-optimal decoding

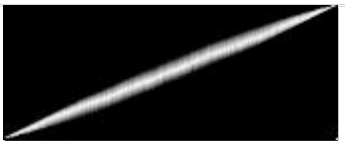


Trellis and complexity issues

- Aggregated trellis [Jégou&Guillemot 05]
- States are defined as

  $(n_k, t_k \bmod T)$
- Decoding with Symbol length constraint
  - Partial information on length is preserved
- Computing cost
  - O(|A|xTxL)
  - Close to optimal decoding





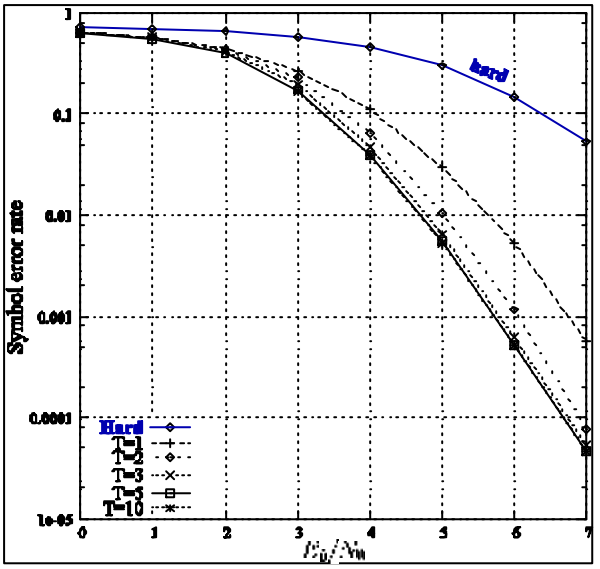
T=100

T= 20

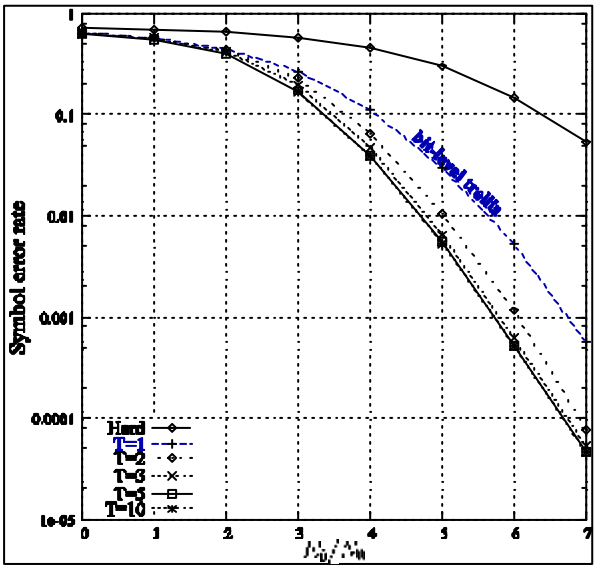
T= 10

•T=1, bit-level trellis
•T=L(S), bit/symbol trellis
•1<T<L(S), trade-off complexity / estimation reliability

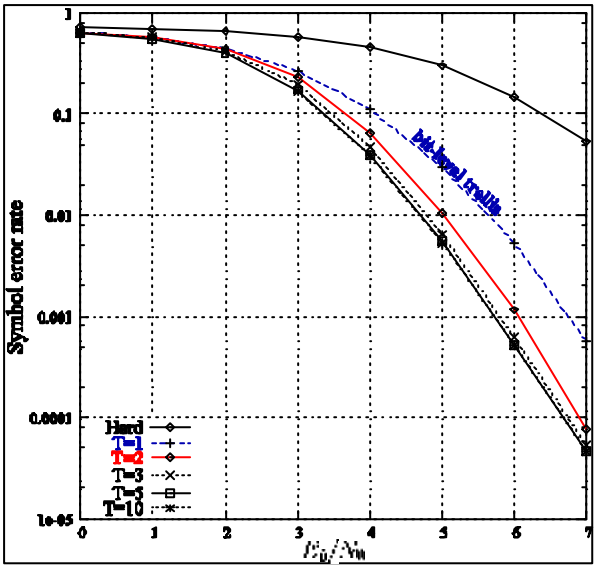
Performance of aggregated state model



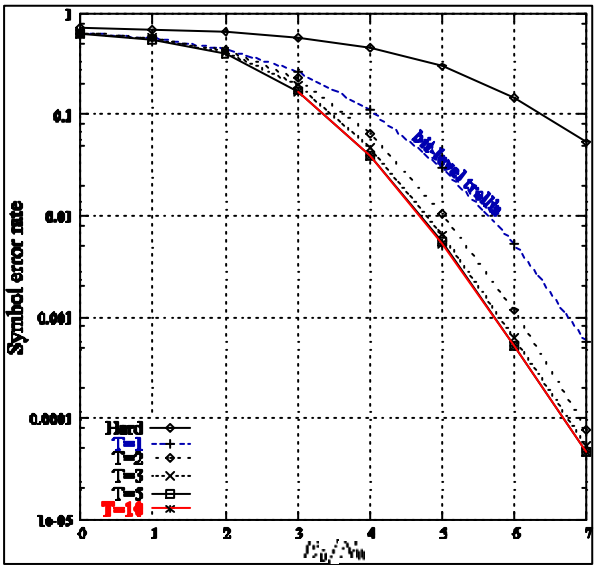
Performance of aggregated state model



Performance of aggregated state model

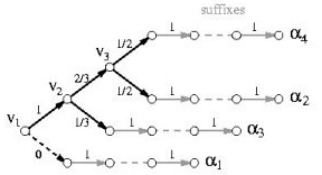


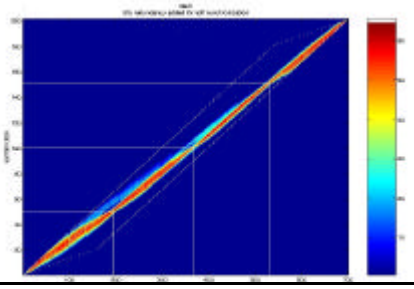
Performance of aggregated state model

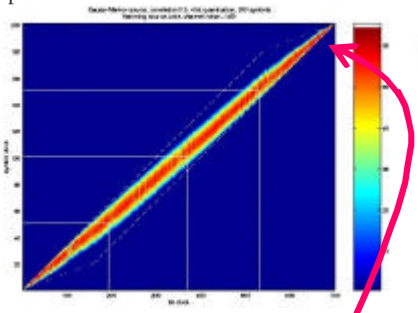


Soft Synchronisation

- To add redundancy specifically dedicated to re-synchronization of bitstream and sequence of symbols
- A-priori known suffix to some symbols at known positions







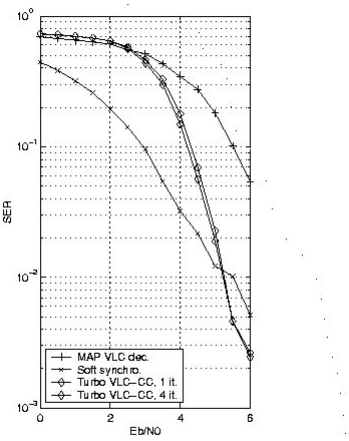
A posteriori Law
p(k,n | measures)

Termination constraint

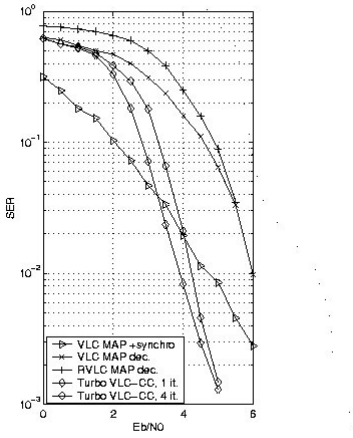
RVLC useless except for a simple decoding

Performance of Soft Synchronisation

ρ = 0.1



ρ = 0.5

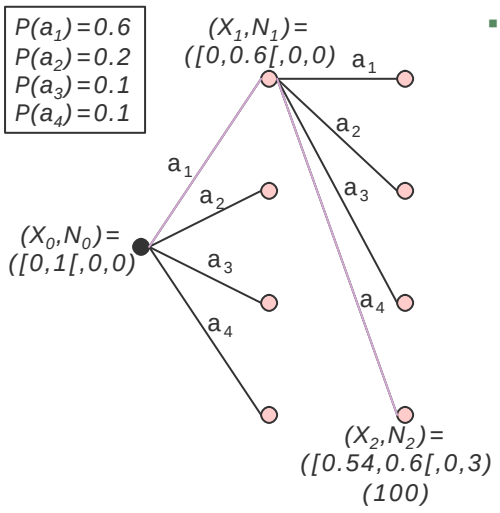


Arithmetic coding

- Arithmetic codes in new standards (H264, JPEG2000)
- Robust arithmetic coding/decoding
  - Augmentation of redundancy
    - Forbidden symbol for error detection & pruning
      [Boyd & al. 97, Sodagar & al. 00, Pettijohn & al. 01…]
    - Forbidden error detection coupled with ARQ
      [Elmasry 99, Chou & Ramchandran 00]
    - coupled with an ECC for error detection [Kozintsev & al. 98, ...]
  - Sequential decoding on decoding tree [Pettijohn & al. 01]

Soft Decoding of Arithmetic Codes

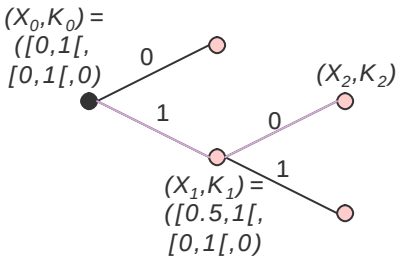
Arithmetic coder : M-ary tree



- Stochastic automaton driven by a symbol clock
  - State of the coding process $X_k = ([l_k , h_k[, nsc_k)$
  - Transitions triggered by a symbol
  - # of bits emitted per transition is random: Model augmented by a counter
  - Transition probabilities follow $P(S_k \mid S_{k-1})$

Soft Decoding of Arithmetic Codes

Arithmetic decoder : Binary tree

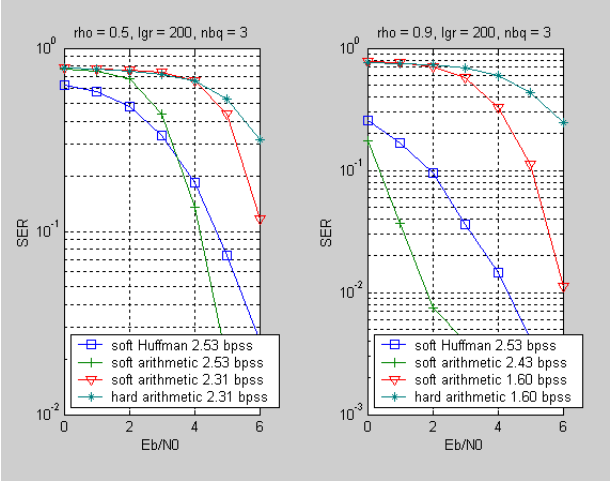


- Stochastic automaton driven by a BIT clock
  - State of the coding process $X_n = ([l_n , h_n[, [L_{Kn} , H_{Kn}[)$
  - Transitions triggered by a BIT
  - # of symbols produced per transition is random: Model augmented by a counter
  - Transition probabilities follow $P(S_{K(n-1)+1} \ldots S_{Kn} \mid S_1^{K(n-1)})$

Soft Decoding of Arithmetic Codes

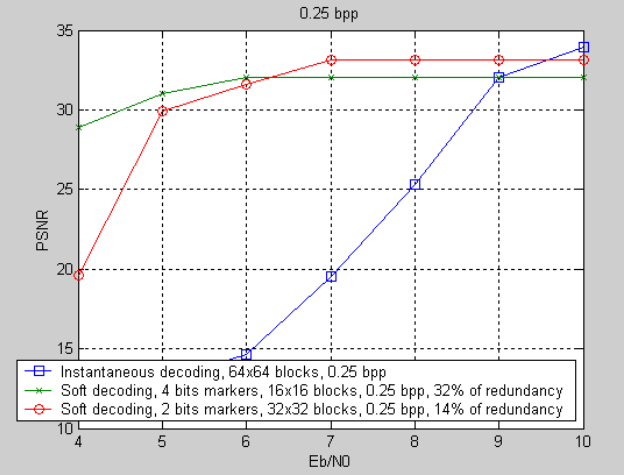
- Soft synchronization
  - Addition of bit patterns : a-priori info.
  - Increase the likelihood of synch. paths
  - Favor the pruning of some branches
  - Redundancy adjusted by size & freq.
- Exponential complexity (tree)
  - Adapted pruning strategy relying on
    - Likelihood of the paths
    - Available computing resources

Results on theoretical sources



Application to JPEG-2000 (EBCOT)

- Headers protected by a convolutional code with rate 1/3
- Possibility of markers addition up to *every stripe*
- Use of standard error detection markers as synchronization markers
- AWGN channel with BPSK modulation
- Results averaged over 100 realizations
- Lena at 0.25 bpp



Results with JPEG-2000 (0.25 bpp; Eb/N0=5 dB)

Without errors
PSNR = 33.98 dB

With error resilience options
Without soft decoding
PSNR = 16.43 dB, 0.92 s.

With error resilience options
With soft decoding, W=10
PSNR = 25.15 dB, 11.16 s.

W=20,
PSNR = 31.91 dB
23.8 s.

Application to CABAC in H.264

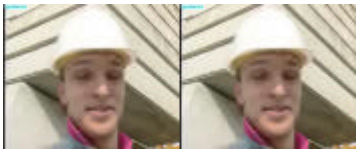
- APP computed for bits in one NAL unit
- Pruning technique
  - Only W paths with maximum probabilities are stored
  - Complexity controlled with W (W=1=> complexity of classical CABAC decoder)
- Termination constraint
  - The estimated sequence is the best path with the right number of transmitted bits and symbols
- With forbidden interval (FI), set of 2D tables storing the range of the LPS accounting for the width of the FI
- The LPS probability space (64 states) remains the same

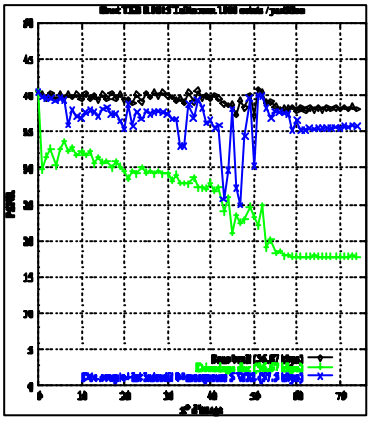
Performance with H.264/CABAC

- Robust CABAC decoder: MAP estimation algorithm
- CABAC options: error detection & correction mechanism

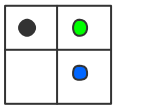
Foreman QCIF@15fps (débit ~56 kbps), BER: 0.0015

- Classical decoding MAP decoding (int 0 W 32 mark 5)



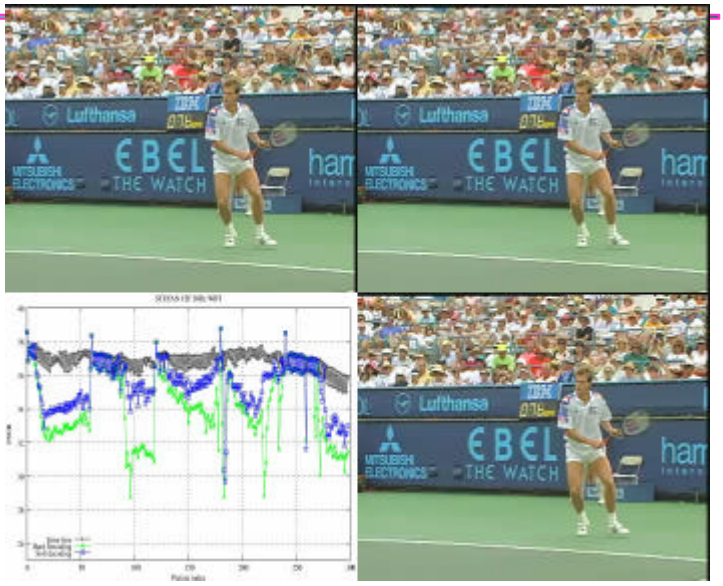


Performance with H.264/CABAC



Pattern WIFI (802.11b)

- Without errors
- Classical decoding
- New mechanism

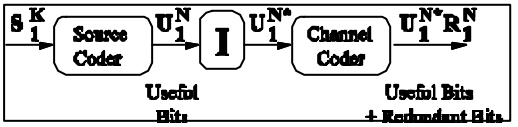


Outline

- Applications and problems addressed
- Soft decoding of VLCs
- Source-channel turbo coding/decoding
- Source codes with properties for source-channel coding
- Source codes with inherent resilience properties
- Conclusion

Source-channel Turbo coding/decoding

- Serial Turbo coding structure

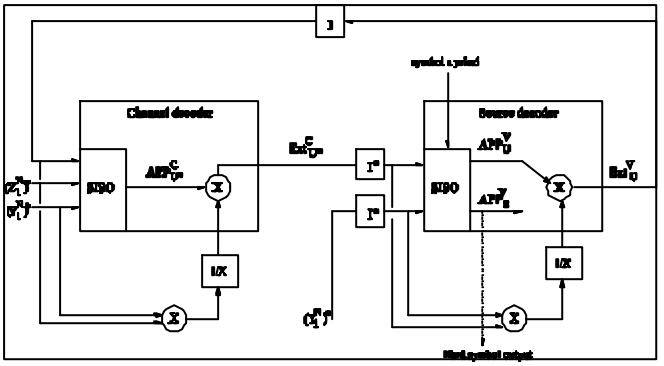


- Direct connection of the 2 HMMs would result in a complex dependency (Bayesian) network with short cycles

  => Not amenable to fast estimation

Source-channel Turbo coding/decoding

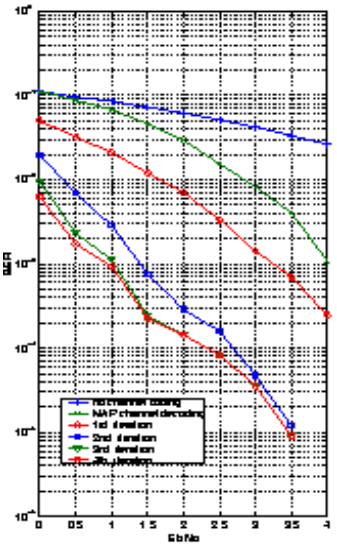
- Serial Turbo decoding structure

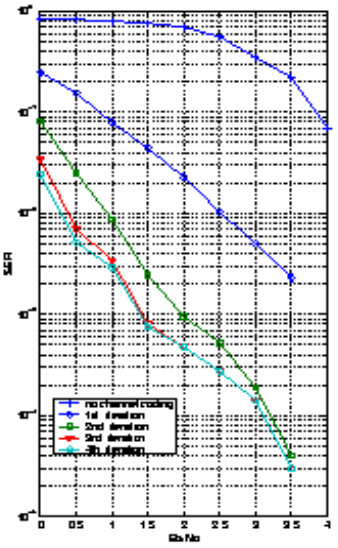


- Extrinsic information: modification induced by a new measure, here all but the local one on the APP of a soft-decoder output

$$Ext^{V}_{U_n}(Y = y|Y_n = y_n) = \frac{\mathbb{P}(U_n|Y = y)}{\mathbb{P}(U_n|Y_n = y_n)\underbrace{Ext^{C}_{U_n}(Y = y|Y_n = y_n)}_{\text{Initialized to 1}}}.$$

Performance illustration of turbo decoding





Outline

- Applications and problems addressed
- Soft decoding of VLCs
- Source-channel turbo coding/decoding
- Source codes with properties for source-channel coding
- Source codes with inherent resilience properties
- Conclusion

New source codes: Motivation

- To attain the capacity of a BSC channel the binary input of the channel should be uniform
- However
  - The output of VLCs is usually not uniform
  - Systematic error correcting codes => the uniformity condition is not satisfied
- Some authors proposed to use non-systematic turbo-codes [Zhy & Alajaji 02,04]
- The problem can be addressed from a source coding perspective

Variable Length Re-writing Systems

- Based on production rules (grammar) of the forms

(symbol,bits)? bits

$$r: \; a\,\bar{l} \rightarrow \bar{b},$$

$$a \in \mathcal{A}, \; \bar{l} \in \{0,1\}^{*}, \bar{b} \in \{0,1\}^{+}$$

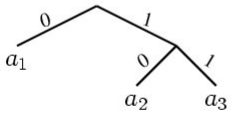
- Extend the usual class of Variable Length Codes
- Small number of states for the encoding/decoding automata (for a given a priori distribution)
- Simple coding/decoding (table look-up)
- Improved joint performance for the joint source/channel decoding with these codes

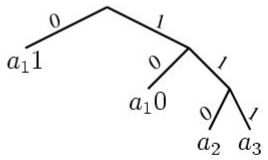
Variable Length Re-writing Systems

- Examples of codes and tree representations:

$$\begin{aligned} r_{1,1}: \quad & a_1 \quad \rightarrow \quad 0 \\ r_{2,1}: \quad & a_2 \quad \rightarrow \quad 10 \\ r_{3,1}: \quad & a_3 \quad \rightarrow \quad 11 \end{aligned}$$

$$\begin{aligned} r_{1,1}: \quad & a_1 1 \quad \rightarrow \quad 0 \\ r_{1,2}: \quad & a_1 0 \quad \rightarrow \quad 10 \\ r_{2,1}: \quad & a_2 \quad \rightarrow \quad 110 \\ r_{3,1}: \quad & a_3 \quad \rightarrow \quad 111 \end{aligned}$$





- Decoder point of view: leaves = a symbol + some bits that are still to be decoded

Variable Length Re-writing Systems

Notation: $a \subset b$ iff a is a prefix of b

1. The set $\bigcup_{i=1}^{|\mathcal{A}|} \bigcup_{j=1}^{|\mathcal{R}_i|} \{\overline{b}_{i,j}\}$ forms a prefix code
2. $\forall i,\ \bigcup_{j=1}^{|\mathcal{R}_i|} \{\overline{l}_{i,j}\}$ is the set $\{\varepsilon\}$ or forms a full prefix code
3. $\forall i\ \forall i'/i' \neq i, \forall j, j',\ \overline{b}_{i,j} = \overline{l}_{i',j'}$ or $\overline{b}_{i,j} \not\subset \overline{l}_{i',j'}$

These conditions are required to ensure that

1. The code is uniquely decodable
2. The code will be uniquely encodable with the proposed backward encoding procedure
3. The bits output by the previous production rule suffice to select the next production rule

Encoding process

- Encoding is processed backward
- Initialization requires the encoding bit 1

$$
\begin{array}{lrcl}
r_{1,1}: & a_1 1 & \rightarrow & 0 \\
r_{1,2}: & a_1 0 & \rightarrow & 10 \\
r_{2,1}: & a_2 & \rightarrow & 110 \\
r_{3,1}: & a_3 & \rightarrow & 111
\end{array}
$$

$$
r_{1,1}: \quad a_1 \quad a_1 \quad a_1 \quad a_1 \quad a_1 \quad 1
$$

Encoding process

- Encoding is processed backward
- Initialization requires the encoding bit 1

$$
\begin{array}{lrcl}
r_{1,1}: & a_1 1 & \rightarrow & 0 \\
r_{1,2}: & a_1 0 & \rightarrow & 10 \\
r_{2,1}: & a_2 & \rightarrow & 110 \\
r_{3,1}: & a_3 & \rightarrow & 111
\end{array}
$$

$$
\begin{array}{lccccccc}
r_{1,1}: & a_1 & a_1 & a_1 & a_1 & a_1 & 1 \\
r_{1,2}: & a_1 & a_1 & a_1 & a_1 & 0
\end{array}
$$

Encoding process

- Encoding is processed backward
- Initialization requires the encoding bit 1

$$
\begin{array}{lllll}
r_{1,1}: & a_1 1 & \rightarrow & 0 \\
r_{1,2}: & a_1 0 & \rightarrow & 10 \\
r_{2,1}: & a_2 & \rightarrow & 110 \\
r_{3,1}: & a_3 & \rightarrow & 111
\end{array}
$$

$$
\begin{array}{llllllll}
r_{1,1}: & a_1 & a_1 & a_1 & a_1 & a_1 & 1 \\
r_{1,2}: & a_1 & a_1 & a_1 & a_1 & 0 \\
r_{1,1}: & a_1 & a_1 & a_1 & 1 & 0
\end{array}
$$

Encoding process

- Encoding is processed backward
- Initialization requires the encoding bit 1

$$
\begin{array}{lllll}
r_{1,1}: & a_1 1 & \rightarrow & 0 \\
r_{1,2}: & a_1 0 & \rightarrow & 10 \\
r_{2,1}: & a_2 & \rightarrow & 110 \\
r_{3,1}: & a_3 & \rightarrow & 111
\end{array}
$$

$$
\begin{array}{llllllll}
r_{1,1}: & a_1 & a_1 & a_1 & a_1 & a_1 & 1 \\
r_{1,2}: & a_1 & a_1 & a_1 & a_1 & 0 \\
r_{1,1}: & a_1 & a_1 & a_1 & 1 & 0 \\
r_{1,2}: & a_1 & a_1 & 0 & 0
\end{array}
$$

Encoding process

- Encoding is processed backward
- Initialization requires the encoding bit 1

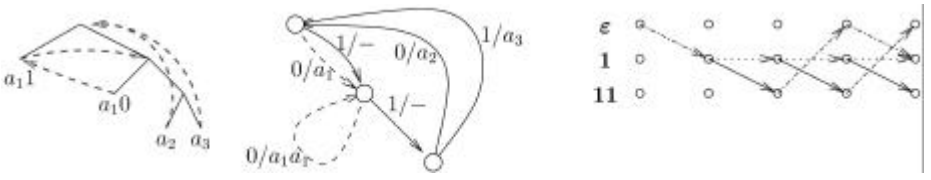
$r_{1,1}$:	$a_1 1$	$\rightarrow$	0
$r_{1,2}$:	$a_1 0$	$\rightarrow$	10
$r_{2,1}$:	a_2	$\rightarrow$	110
$r_{3,1}$:	a_3	$\rightarrow$	111

$r_{1,1}$:	a_1	a_1	a_1	a_1	a_1	1
$r_{1,2}$:	a_1	a_1	a_1	a_1	0	
$r_{1,1}$:	a_1	a_1	a_1	1	0	
$r_{1,2}$:	a_1	a_1	0	0		
$r_{1,1}$:	a_1	1	0	0		

- Emitted sequence: 000

Variable Length Re-writing Systems

- Encoding and decoding can be implemented with sequential automata
- Definition of internal states
  - Bit sequence of variable length
  - Encoder: the smallest set of bit sequences that suffice to choose any rule
  - Decoder: a set of bits that do not generate a rule
- Example: tree, decoding automaton and trellis



Compression efficiency

- Compression efficiency: Expected number of bits produced by a production rule

$$\sum_{r_{i,j}} \delta_{i,j}\, \mathbb{P}(R_t = r_{i,j})$$

$$\delta_{i,j} = L(\overline{b}_{i,j}) - L(\overline{l}_{i,j})$$

- $(R_{L(S)}, \ldots, R_t, R_{t-1})$ forms a Markov chain of transition probabilities

$$\mathbb{P}(R_t = r_{i,j} | R_{t+1} = r_{i',j'}) = \begin{cases} \mathbb{P}(a_i) & \text{if } \overline{l}_{i,j} \subset \overline{b}_{i',j'} \\ 0 & \text{otherwise.} \end{cases}$$

Compression efficiency

Assuming that $(R_t)_t$ is ergodic, $\lambda = \mathbb{P}(R_t)$ is obtained from $\pi = \mathbb{P}(R_t | R_{t+1})$ as the solution of the matricial equation

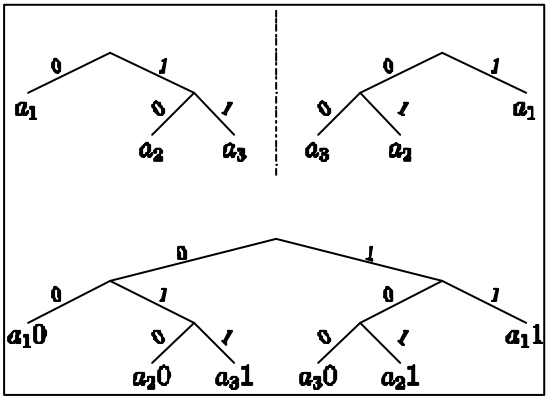
$$\lambda = \pi \times \lambda.$$

$\Rightarrow$ it leads to $E(L(\overline{B}_{i,j}) - L(\overline{L}_{i,j}))$, i.e. to the asymptotic compression efficiency

Example:
VLRS $\{a_1 1 \to 0, a_1 0 \to 10, a_2 \to 110, a_3 \to 111\}$,
$\mathbb{P}(a_i) = \{0.7, 0.2, 0.1\} \Rightarrow$ expected length $= 1.188$ bits.
Huffman codes $\Rightarrow 1.3$ bits.
In average, a_1 is coded with less that 0.5 bits.

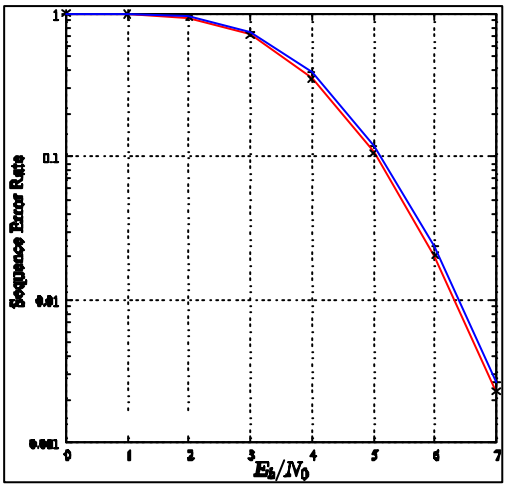
Mirrored construction

- To design a code leading to a uniform distribution of bits
- Construction from a VLC (same EDL)



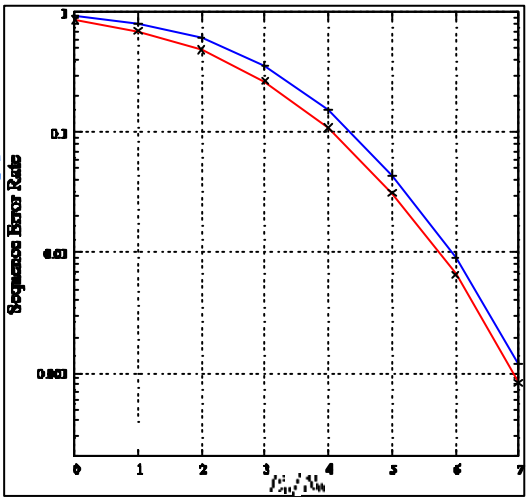
Soft decoding performance

- Code {00, 11, 010, 101, 0110}
- P(0)=0.6
- Mirror VLRS vs RVLC
- Balakirsky-like state model
- Viterbi decoding



Soft decoding performance

- Code {00, 11, 010, 101, 0110}
- P(0)=0.917
- Mirror VLRS vs RVLC
- Balakirsky-like state model
- Viterbi decoding



Outline

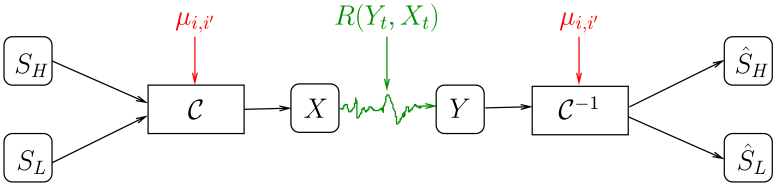
- Applications and problems addressed
- Soft decoding of VLCs
- Source-channel turbo coding/decoding
- Source codes with properties for source-channel coding
- Source codes with inherent resilience
- Conclusion

Multiplexed Codes, principle (1)

- **Most of existing JSCC solutions introduce some redundancy in the encoded bitstream**
- **A new family of codes:**
  - **no redundancy**
  - **reduction of the impact of channel noise**
  - **assume the existence of two sources (S_H and S_L)**
  - **guarantee no error propagation for S_H (synchronous)**

  ® **Inherent Unequal Error Protection (UEP)**
- **Design Principle :**
  - **redundant FLCs designed for S_H**
  - **redundancy used to describe S_L**

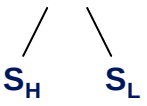
Transmission scheme: 2 sources



- S_H: First-order Markov source, of high priority
  - Takes its values on a finite alphabet $\boldsymbol{A=\{a_1 \ldots a_i \ldots\}}$
  - Transitions probabilities $\mu_{i,i'} = P(S_t=a_i / S_{t-1}=a_{i'})$
- S_L: uniform memoryless binary source of low priority
- Channel: Discrete Memoryless Channel
  - $R(Y_t,X_t)=P(Y_t$ received / X_t emitted)

Multiplexed Codes, principle (2)

- Alphabet A of S_H: $a_1 \ldots a_i \ldots a_W$, cardinality W
- Codewords of length c
  - Þ set X of 2^c codewords
- X partionned in W equivalence classes associated to symbols of S_H: $C_1, \ldots C_i, \ldots C_W$
  Let N_i = card(C_i)
- Bijection: X ® A x [0..N_i-1]
  cwd ® (a_i, q)



Stationary multiplexed codes

- Redundant FLC
- C_i: equivalence class of a_i
- q_i: index of a codeword within its equiv. class
- Inherits FLC properties
  - Strict sense synchronization
  - Random data access

a_i	μ_i	codeword	index q_i	
a_1	0.4	000	0	C_1
		001	1	
		010	2	
a_2	0.2	011	0	C_2
		100	1	
a_3	0.2	101	0	C_3
a_4	0.1	110	0	C_4
a_5	0.1	111	0	C_5

Compression efficiency

- For each codeword:
  - c bits used
  - index variable Q_i : $\log_2(N_i)$ bits for S_L

  Þ a symbol of S_H is coded with $c-\log_2(N_i)$ bits

- Mean description length (mdl) of S_H:

$$- \sum m_i \log_2(N_i / 2^c)$$

- N_i are chosen to reach the entropy, i.e closest to

$$N_i = m_i 2^c$$

Multiplexing capacity

- Number L of sequences representating a given sequence s_H of length K:

$$L = \tilde{O}_{t=1..K} N_t$$

- Theorical multiplexing capacity: $\log_2(L)$ bits

- How to use this capacity?

  ® conversion of S_L into a sequence of states q_t (0..N_i-1)

- Assumption:

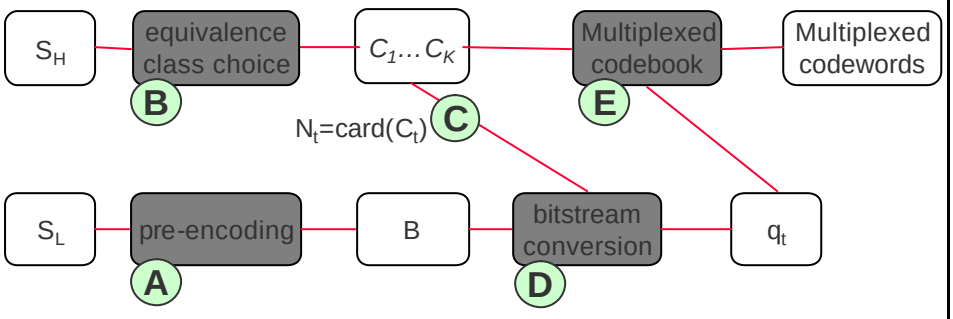
  S_L already encoded as a bitstream b

Multiplexing capacity

s_H	a_1	a_4	a_5	a_2	a_3	a_3	a_1	a_2
	000	110	111	011	101	101	000	011
C_i	001			100			001	100
	010						010	
$\lvert C_i \rvert$	3	1	1	2	1	1	3	2

- Number of sequences representing s_H: 3*2*3*2=36
- Multiplexing capacity: $\log_2(36)=5.17$ bits
- s_H is finally coded with 3*8-$\lvert 5.17 \rvert$=19 bits

Encoding scheme (1)



Encoding scheme (2)

A S_L is pre-encoded as a bitstream b (compressed)

B " t, the realization s_t of S_H indexes an equivalence class

C Processing N_t provides the cardinality of the index variable Q_t

D The bitstream b is converted into a sequence of N_t-valued variables

E Couples (a_1, q_t) provides entries in the multiplexed table

Step D: conversion of b (1)

- First approach:
  - b ° binary representation of an integer g
  - g : realization of a L-valued variable, indexing codewords of the whole sequence for a realization
  - g is transformed into a sequence of state (q_t) using Euclidean divisions by N_t

$$g = q_1 + N_1 (q_2 + N_2 (\dots + N_t (q_{t+1} + \dots N_{K-1} q_K) \dots) \dots))$$

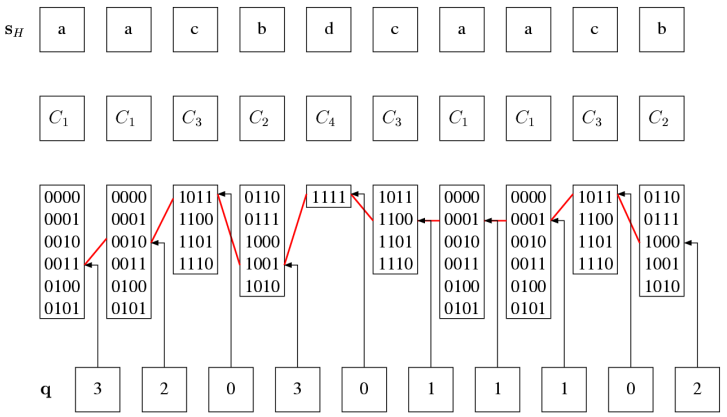
- Main Advantage & Drawback
  - Optimality in terms of compression (depending on c)
  - but dealing with long integers

Step D : conversion of b (2)

- **2nd approach:**
  - **no calculation of a global state**
  - **states (q_t) are processed using transformations with low computing cost**
  - **Not optimal in compression: overhead < 1% to entropy**
- **Binary multiplexed codes:**
  - **Bits of b directly indexe codewords**
  - **$N_i=2^{li}$**
- **MUX-codes constructed from VLC codetrees**
  - **SER(BER) = BER . *mdl* + O(BER) on a BSC**

Step D: Conversion of S_L: problem statement

- **Several codeword sequences are available**
  ® bijection with a sequence of N_t-valued states (q_t)



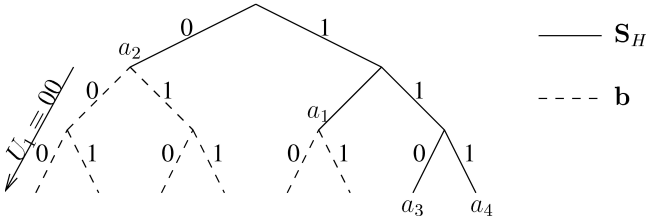
Binary multiplexed codes (1)

- **Is it possible to avoid step D?: Yes, Binary Multiplexed Codes**
- **Definition :**
  - **a multiplexed code is binary iff "i, N_i is a power of 2**
- **Consequences :**
  - **codewords in a given equivalence class can be directly indexed a segment U_i of $log(N_i)$ bits**

  ↓**no bitstream conversion is required**
- **Compression efficiency of Huffman codes**

Construction from a VLC codetree

- **The VLC codetree is completed to reach the longest codeword length**
- **A codeword prefix is used for S_H**
- **A codeword suffix "store" some bits U_t of B**



- **SER performance on a BSC :**

  SER(BER) = BER . m*dl*+ O(BER)

Binary multiplexed codes (2)

class $\mathcal{C}_i$	cwd $c_{i,q}$	a_i	N_i	$\mathbb{P}(a_i)$	index q	*or* U_i
$\mathcal{C}_1$	000	a_1	2	0.30	0	0
	001				1	1
$\mathcal{C}_2$	010	a_2	4	0.43	0	00
	011				1	01
	100				2	10
	101				3	11
$\mathcal{C}_3$	110	a_3	1	0.25	0	∅
$\mathcal{C}_4$	111	a_4	1	0.02	0	∅

Encoding example

s_H = $a_1\ a_2\ a_2\ a_3\ a_2\ a_1\ a_2\ a_4$
b = **0101010101**

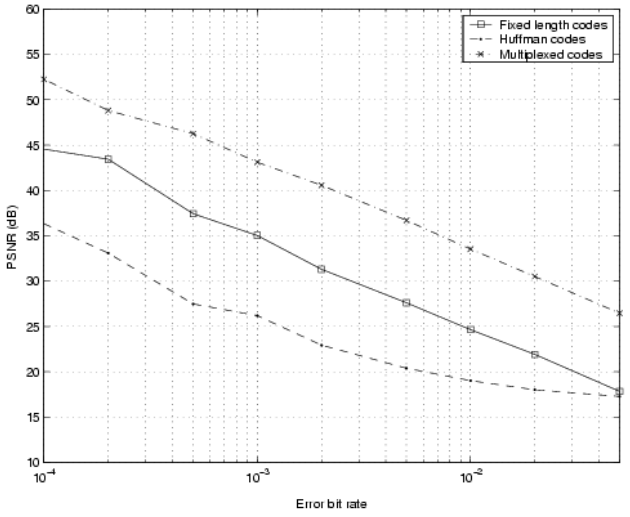
Þ

$log(N_i)_{t=1..8}$ = 1 2 2 0 2 1 2 0
$(U_t)_{t=1..8}$ = (0, 10, 10, Æ, 10, 1, 01, Æ)

Using the multiplexed codebook, the generated bitstream is

000 100 100 110 100 001 011 111

Multiplexed Codes

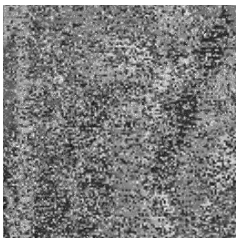


FLC

Huf

Mul

Simulation results (BSC)

	FLC (uncompressed)	Huffman	MUX codes
BER= 0.01			
BER= 0.05			

Conclusion on Stationary MUX-codes

- **De-synchronization confined to the low priority source**
- **Source UEP**
- **Better SER and SNR results than FLCs for S_H**
- **Random data access for S_H**
- **Low complexity soft decoding for S_H**

Is it possible to exploit higher-order source statistics?

First-order multiplexed codes

- **Motivation: to exploit the conditional probabilities $m_{i,i'}$**
- **Code construction**
  - **$\forall a_i$?** ***A, a multiplexed code is designed for the conditional pdf $m_{i,i'}$***
  - ***Equivalence classes indexed by $N_i^{i'}$***
- ***For a stationary discrete first-order Markov source, the conditional entropy is reached iff***

$$\forall (a_i, a_{i'}) \in A^2 \; |N_i^{i'}| = P(a_i / a_{i'}) \; |X|$$

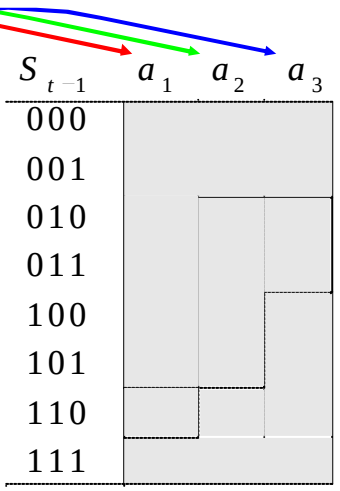
Example

$\mu_{i,i'}$	a_1	a_2	a_3
a_1	3	1	1
	4	4	4
a_2	1	1	1
	8	2	4
a_3	1	1	1
	8	4	2

$entropy_0 = 1.5$

$entropy_1 = 1.28$

$mdl = 1.28$



Index Assignment (IA)

- A symbol S_t is not correctly reconstructed in case of
  - Error in the codeword itself
  - Error in the previous reconstructed symbol
    => Error propagation may occur on contexts
- The IA conditioned by a given context has to be optimized by taking into account the other contexts
- Some codewords may represent the same symbol for all the previous conditioning values
  ? The set of codewords that have this property is called kernel

Kernel

- **Definition: the kernel is the set of codewords** k =

  $\{ x \in \mathcal{X} / \exists a_i \in A, \forall a_{i'} \in A, x \in C_i^{i'} \}$

- ***In the example,*** **| k |=3**
- ***It could be 4***

S_{t-1}	a_1	a_2	a_3
000	a_1	a_1	a_1
001	a_1	a_1	a_1
010			
011			
100			
101			
110			
111	a_3	a_3	a_3

Kernel

- **Definition: the kernel is the set of codewords k =**

  $\{ x \in \mathcal{X} / \exists a_i \in A, \forall a_{i'} \in A, x \in C_i^{i'} \}$

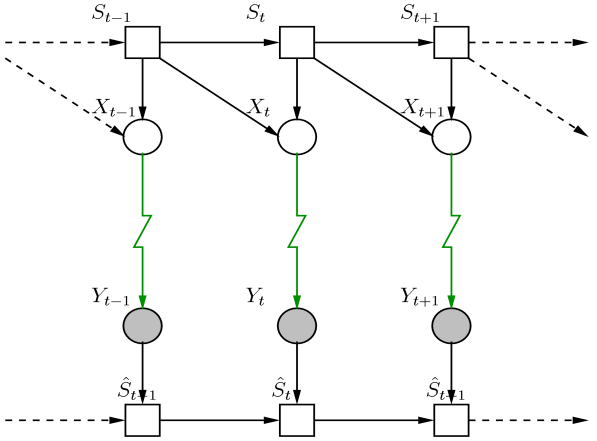
- ***In the example,*** **| k |=3**
- ***It could be 4***

S_{t-1}	a_1	a_2	a_3
000	a_1	a_1	a_1
001	a_1	a_1	a_1
010	a_2	a_2	a_2
011			
100			
101			
110			
111	a_3	a_3	a_3

The kernel offers natural hard synchronization properties

Error resilience analysis (S_H)

- **The product model (S_t,$\hat{S}_t$) is a Markov process**



Error resilience analysis

- **Transition probabilities of this model given by**

$$P(\hat{S}_t = a_j, S_t = a_i / \hat{S}_{t-1} = a_{j'}, S_{t-1} = a_{i'}) = \frac{\mu_{i,i'}}{N_i^{i'}} \sum_{(X_t, Y_t) \in C_i^{i'} \times C_j^{j'}} R(Y_t, X_t)$$

- **If this matrix positive, the asymptotic marginal probability $P(S_t=a_i,\check{S}_t=a_j)$ can be computed from this matrix (Perron-Frobenius Theorem) (normalized eigenvectors associated to the eignevalue 1)**

$$SER_{K \to \infty} = 1 - \sum_{a_i \in A} P(\hat{S}_t = a_i, S_t = a_i)$$

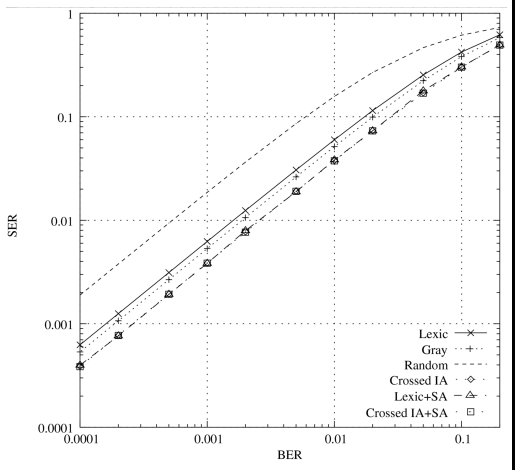
$$MSE_{K \to \infty} = \sum_{(a_i, a_j) \in A^2} P(\hat{S}_t = a_j, S_t = a_i)(a_j - a_i)^2$$

Cross-Index Assignment

- **Objective:**
  optimizing the analytical expressions of SER and SNR
- **Efficient optimization algorithms in the litterature:**
  - **Binary Switching Algorithm (BSA) [Zeger & al. 90]**
  - **Simulated Annealing (SA), e.g. [Farvadin & al. 91]**
- **For the SER criteria, Simulated Annealing turns out to maximize the cardinal of the kernel**

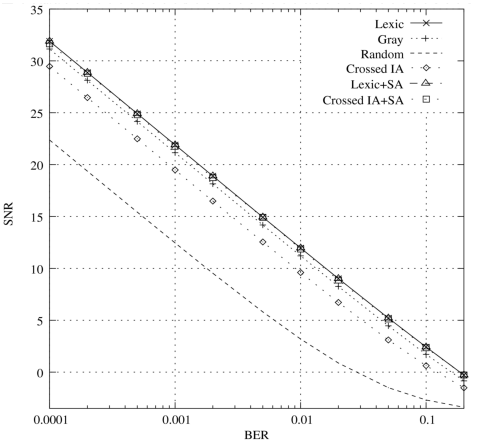
Symbol Error Rate on a BSC

- **S_H: Gauss-Markov source of correlation ? =0.5**
- **8 quantization cells**
- **$H_1(S_H)$=2.29**
- **Binary symmetrical channel**
- **1st-order MUX-code:**
  - **c=5**
  - **mdl=2.35**



Signal to Noise Ratio on a BSC

- **S_H: Gauss-Markov source of correlation ? =0.5**
- **8 quantization cells**
- **$H_1(S_H)$=2.29**
- **Binary symmetrical channel (BSC)**
- **1st-order MUX-code:**
  - **c=5**
  - **mdl=2.35**



Compression assessment in SVC

- Texture residue coded in CAVLC
  - High priority *NumTrail (*4 VLC tables depending on context)
  - Low priority *TrailingOnes* sign, *RunBefore*, *TotalRun*

	Tab 1	Tab 2	Tab 3	Tab 4
Entropy	2.1543	3.3809	4.6972	5.4692
Mdl cavlc	2.1620	3.4062	4.7749	6 (FLC)
Mdl huffman	2.1608	3.4062	4.7338	5.5040
Mdl multiplexed codes	**2.1548**	**3.3871**	**4.6972**	**5.4693**
Symbols distribution	62%	25%	10%	1%

- No loss in compression efficiency: error resilience for free!!

Conclusion

- Significant gains in SER/PSNR in exploiting existing VLC sub-optimality
  - High gains if soft measures available in the application layer
- Extra gain in keeping/inserting redundancy in the chain
  - A priori source information
  - Error correcting code
  - This has a cost in bandwidth
- Codes with inherent resilience
  - Almost no loss in compression efficiency
- All this makes “lite” mechanisms in lower layers meaningful to save bandwidth