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

<table>
<thead>
<tr>
<th>Layer</th>
<th>UDP</th>
<th>UDP-lite</th>
<th>UDP</th>
<th>UDP-lite</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean number of received packets</td>
<td>6422</td>
<td>9466</td>
<td>9751</td>
<td>9887</td>
</tr>
<tr>
<td>% Mean gain</td>
<td>47.4</td>
<td>1.4</td>
<td>6.9</td>
<td></td>
</tr>
</tbody>
</table>

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 chain and the sub-optimality of the code.

$W_i \rightarrow W_N$

Modelling the dependencies between the processes of the chain with a Bayesian network formalism
Modelling the VLC process

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

\[ X_n = (v_n) \]

- The sequence of states \( X_n \) forms a Markov chain

Soft Decoding of VLCs

- Bit clock state model and estimator (memoryless source)

  \[ \hat{X}_1, \ldots, \hat{X}_n = \arg \max_{X_1, \ldots, X_n} P(X_1, \ldots, X_n | Y_1, \ldots, Y_n) \]

  \[ \Rightarrow \text{SOVA algorithm [Hagenauer & Hoeher 89]} \]

- MAP estimator

  \[ \hat{X}_n = \arg \max_{X_n} P(X_n, Y_1, Y_2, \ldots, Y_n) \]

  \[ \Rightarrow \text{Two recursions BCJR algorithm [Bahl, Cocke, Jelinek, Raviv, 97]} \]

- MPM estimator

  \[ \hat{X}_n = \arg \max_{X_n} P(X_n | Y_1, Y_2, \ldots, Y_n) \]

  \[ \Rightarrow \text{BCJR algorithm [Bahl, Cocke, Jelinek, Raviv, 97]} \]
Soft Decoding of VLCs

- To force an additional termination constraint

\[(X_n, K_n) = (v_n, k)\]

*Augmented Markov chain*

Constraint on the # of symbols, \(K_n = K\)

Noisy measurements

\[\Rightarrow \text{Bit/symbol trellis [Hagenauer 00]}\]

- 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

\[\Rightarrow \text{Augmented bit/symbol trellis}\]

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

\[ \Rightarrow \text{State space of reduced dimension} \]

\[ \Rightarrow \text{But sub-optimal (inter-symbol correlation not exploited)} \]

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

\[ \Rightarrow \text{Need for a symbol clock Markov model of the sequence of symbols} \]

---

**Symbol clock model:** the simple case of FLCs:

- Fast estimation algorithms - MAP or MPM – on \((S_k)\)
  \[ \hat{S}_k = \arg \max_{S_k} P(S_k|Y_1, Y_2, \cdots Y_k) = \arg \max_{S_k} P(S_k|\overline{Y}_1, \overline{Y}_2, \cdots \overline{Y}_k) \]

Two recursions algorithm (BCJR algorithm)

\[
P(S_k|Y_k) \propto P(Y_k|S_k) \sum_{S_{k+1}} P(S_{k+1}|S_k)P(S_k|Y_k^{k-1})
\]

\[
P(\overline{Y}_k|S_k) = P(\overline{Y}_k|U_1)
\]

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:
    \[
    P(\bar{Y}_i|S_k, N_k) = P(\bar{Y}_{N_{k+1}}, \bar{U}_{N_{k+1}})
    \]
  - Joint problem of segmentation and estimation
    \[
    N_k = N_{k-1} + L(S_k)
    \]

1- **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 $(S_k, N_k) \Rightarrow$ Clock conversion on soft information, given the recursion $N_k = N_{k-1} + \text{length}(S_k)$

3- To incorporate the inter-symbol correlation
   \[
   P(\bar{Y}_i|S_k, N_k) \propto P(\bar{Y}_i^{N_k}, \bar{U}_i^{N_k})
   \]
   \[
   P(S_i|\bar{Y}_i) \propto P(S_i|\bar{Y}_i^{k}), P(S_i^{k-k} | S_i)
   \]
   \[
   P(S_i|\bar{Y}_i^{k}) \propto P(S_i^{k-k}) \sum_{S_i^{k-k}} P(S_i|S_i^{k-k}) P(S_i^{k-k} | \bar{Y}_i^{k-1})
   \]

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
  - $O(|A| \times L)$
- 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
  - $O(|A| \times L^2)$
  - Sub-optimal decoding
Trellis and complexity issues

- Aggregated trellis [Jégou & Guillemot 05]
- States are defined as
  \((n_k, t_k \mod T)\)
- Decoding with Symbol length constraint
  - Partial information on length is preserved
- Computing cost
  - \(O(|A| \times T \times L)\)
  - Close to optimal decoding
- \(T=1\), 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

![Graph](image)

Performance of aggregated state model

![Graph](image)
Performance of aggregated state model

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

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
Performance of Soft Synchronisation

\[ \rho = 0.1 \quad \rho = 0.5 \]

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 $P(S_k | S_{k-1})$

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 $P(S_{K(n-1)+1} … S_{Kn} | S_{K(n-1)})$
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)

<table>
<thead>
<tr>
<th>Without errors</th>
<th>With error resilience options</th>
<th>With error resilience options</th>
<th>W=20, PSNR = 31.91 dB</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR = 33.98 dB</td>
<td>Without soft decoding</td>
<td>With soft decoding, W=10</td>
<td>PSNR = 25.15 dB, 11.16 s.</td>
</tr>
<tr>
<td>PSNR = 16.43 dB, 0.92 s.</td>
<td>PSNR = 31.91 dB, 23.8 s.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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

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

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

$$Ex_{i_{in}}^k(Y = y | Y_u = y_u) = \frac{P(U_u | Y_u = y_u) \cdot \hat{E} x^k_{i_{in}}(Y = y | Y_u = y_u)}{P(U_u | Y_u = y_u)}$$

Initialized to 1
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:

\[ r_{1,1} : a_1 \rightarrow 0 \]
\[ r_{1,2} : a_1 0 \rightarrow 10 \]
\[ r_{2,1} : a_2 \rightarrow 10 \]
\[ r_{2,2} : a_2 \rightarrow 110 \]
\[ r_{3,1} : a_3 \rightarrow 11 \]
\[ r_{3,2} : a_3 \rightarrow 111 \]

- 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 \( \bigcup_{i=1}^{A} \bigcup_{i=1}^{R_i} \{ \overline{b}_{i,j} \} \) forms a prefix code
2. \( \forall i, \bigcup_{j=1}^{\overline{R}_i} \{ \overline{b}_{i,j} \} \) is the set \( \{ \epsilon \} \) or forms a full prefix code
3. \( \forall i \forall j' / j' \neq i, \forall j, j', \overline{b}_{i,j} \approx l_{y,j'} \) or \( \overline{b}_{i,j} \not\subset l_{y,j'} \)

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

\[
\begin{align*}
  r_{1,1} : a_1 &\rightarrow 0 \\
  r_{1,2} : a_1 &\rightarrow 10 \\
  r_{2,1} : a_2 &\rightarrow 110 \\
  r_{3,1} : a_3 &\rightarrow 111
\end{align*}
\]

\[
\begin{align*}
  r_{1,1} : a_1 a_1 a_1 a_1 a_1 &\rightarrow 0 \\
  r_{1,2} : a_1 &\rightarrow 10 \\
  r_{2,1} : a_1 a_1 a_1 &\rightarrow 110 \\
  r_{3,1} : a_1 a_1 &\rightarrow 111
\end{align*}
\]
Encoding process

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

\[
\begin{array}{c|c}
  r_{1,1} : a_1 & 1 \\
r_{1,2} : a_0 & 10 \\
r_{2,1} : a_2 & 110 \\
r_{3,1} : a_3 & 111 \\
\end{array}
\]

\[
\begin{array}{c|c}
  r_{1,1} : a_1 a_1 a_1 a_1 a_1 & 1 \\
r_{1,2} : a_1 a_1 a_1 a_1 & 0 \\
r_{1,1} : a_1 a_1 a_1 & 0 \\
r_{1,2} : a_1 & 0 \\
\end{array}
\]
Encoding process

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

\[
\begin{align*}
r_{1,1} : a_1^1 & \to 0 & r_{1,1} : a_1 a_1 a_1 a_1 a_1 1 \\
r_{1,2} : a_1 0 & \to 10 & r_{1,2} : a_1 a_1 a_1 a_1 0 \\
r_{2,1} : a_2 & \to 110 & r_{1,1} : a_1 a_1 a_1 1 0 \\
r_{3,1} : a_3 & \to 111 & r_{1,2} : a_1 a_1 0 0 \\
\end{align*}
\]

- 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

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

\[ \delta_{i,j} = L(\bar{b}_{i,j}) - L(\bar{l}_{i,j}) \]

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

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

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

\[ \lambda = \pi \times \lambda. \]

⇒ it leads to \(E(L(\bar{b}_{i,j}) - L(\bar{l}_{i,j}))\), i.e. to the asymptotic compression efficiency

Example:

| VLRS | a₁₁ → 0, a₁₀ → 10, a₂ → 110, a₃ → 111 | \(\mathbb{P}(a_i) = \{0.7, 0.2, 0.1\}\) ⇒ expected length = 1.188 bits. Huffman codes ⇒ 1.3 bits.
|      | In average, a₁ 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)

![Diagram of mirrored construction]

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 graph]
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 $A=\{a_1...a_n\}$
  - Transitions probabilities $\mu_{i,j} = P(S_t=a_i \mid S_{t-1}=a_j)$
- $S_L$: uniform memoryless binary source of low priority
- Channel: Discrete Memoryless Channel
  - $R(Y_t,X_t)=P(Y_t \text{ received} \mid X_t \text{ emitted})$
**Multiplexed Codes, principle (2)**

- **Alphabet** $A$ of $S_H$: $a_1...a_i...a_W$, cardinality $\Omega$
- **Codewords of length $c$**
  - set $X$ of $2^c$ codewords
- **$X$** partitioned in $\Omega$ equivalence classes associated to symbols of $S_H$: $C_1, ...C_i, ...C_\Omega$
  - Let $N_i = \text{card}(C_i)$
- **Bijection:**
  - $X \rightarrow A \times [0..N_i-1]$
  - $\text{cwd} \rightarrow (a_i,q_i)$

**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

<table>
<thead>
<tr>
<th>$a_i$</th>
<th>$\mu_i$</th>
<th>codeword</th>
<th>index $q_i$</th>
<th>$C_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td>0.4</td>
<td>000</td>
<td>0</td>
<td>$C_1$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>001</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>010</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$a_2$</td>
<td>0.2</td>
<td>011</td>
<td>0</td>
<td>$C_2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$a_3$</td>
<td>0.2</td>
<td>101</td>
<td>0</td>
<td>$C_3$</td>
</tr>
<tr>
<td>$a_4$</td>
<td>0.1</td>
<td>110</td>
<td>0</td>
<td>$C_4$</td>
</tr>
<tr>
<td>$a_5$</td>
<td>0.1</td>
<td>111</td>
<td>0</td>
<td>$C_5$</td>
</tr>
</tbody>
</table>
Compression efficiency

- For each codeword:
  - $c$ bits used
  - Index variable $Q_i$ : $\log_2(N_i)$ bits for $S_L$
    $\Rightarrow$ a symbol of $S_H$ is coded with $c \cdot \log_2(N_i)$ bits

- Mean description length (mdl) of $S_H$:
  $- \sum \mu_i \log_2(N_i / 2^c)$

- $N_i$ are chosen to reach the entropy, i.e. closest to $N_i = \mu_i 2^c$

Multiplexing capacity

- Number $\Lambda$ of sequences representing a given sequence $s_H$ of length $K$:
  $\Lambda = \prod_{t=1..K} N_t$

- Theoretical multiplexing capacity: $\log_2(\Lambda)$ bits

- How to use this capacity?
  $\Rightarrow$ conversion of $S_L$ into a sequence of states $q_t(0..N_t-1)$

- Assumption:
  $S_L$ already encoded as a bitstream $b$
Multiplexing capacity

<table>
<thead>
<tr>
<th>$s_H$</th>
<th>$a_1$</th>
<th>$a_4$</th>
<th>$a_5$</th>
<th>$a_2$</th>
<th>$a_3$</th>
<th>$a_3$</th>
<th>$a_1$</th>
<th>$a_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_i$</td>
<td>000</td>
<td>110</td>
<td>111</td>
<td>011</td>
<td>101</td>
<td>101</td>
<td>000</td>
<td>011</td>
</tr>
<tr>
<td></td>
<td>001</td>
<td></td>
<td></td>
<td>100</td>
<td></td>
<td></td>
<td>001</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>010</td>
<td></td>
<td></td>
<td>010</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$</td>
<td>C_i</td>
<td>$</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

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

Encoding scheme (1)
Encoding scheme (2)

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

B. \( \forall t \), 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 \((a_t, q_t)\) provides entries in the multiplexed table

Step \( D \): conversion of \( b \) (1)

- First approach:
  - \( b \equiv \) binary representation of an integer \( \gamma \)
  - \( \gamma \) : realization of a \( \Lambda \)-valued variable, indexing codewords of the whole sequence for a realization
  - \( \gamma \) is transformed into a sequence of state \((q_t)\) using Euclidean divisions by \( N_t \)
    \[
    \gamma = q_1 + N_1 (q_2 + N_2 ( ... + N_t (q_{t+1} + ... N_{K-1} q_K) ... ))
    \]

- 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 index codewords
  - $N_l=2^l$

- MUX-codes constructed from VLC codetrees
  - $\text{SER(BER)} = \text{BER \cdot mdl} + O(\text{BER})$ on a BSC

Step D: Conversion of $S_L$: problem statement

- Several codeword sequences are available
  → bijection with a sequence of $N_l$-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 $\forall i, N_i$ 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 $U_i$ of $B$

SER performance on a BSC:

$\text{SER(BER)} = \text{BER} \cdot m d l + O(\text{BER})$
Binary multiplexed codes (2)

<table>
<thead>
<tr>
<th>class $C_i$</th>
<th>cwd</th>
<th>$c_{i,q}$</th>
<th>$a_i$</th>
<th>$N_i$</th>
<th>$\mathbb{P}(a_i)$</th>
<th>index $q$</th>
<th>$or\ U_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_1$</td>
<td>000</td>
<td>001</td>
<td>$a_1$</td>
<td>2</td>
<td>0.30</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$C_2$</td>
<td>010</td>
<td>011</td>
<td>$a_2$</td>
<td>4</td>
<td>0.43</td>
<td>0</td>
<td>00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>01</td>
</tr>
<tr>
<td></td>
<td></td>
<td>101</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>$C_3$</td>
<td>110</td>
<td>$a_3$</td>
<td>1</td>
<td>0.25</td>
<td></td>
<td>0</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$C_4$</td>
<td>111</td>
<td>$a_4$</td>
<td>1</td>
<td>0.02</td>
<td></td>
<td>0</td>
<td>$\emptyset$</td>
</tr>
</tbody>
</table>

Using the multiplexed codebook, the generated bitstream is

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

$$\Rightarrow$$

$$\log(N_{t=1..8}) = 1 \ 2 \ 2 \ 0 \ 2 \ 1 \ 2 \ 0$$
$$(U_t)_{t=1..8} = (0, 10, 10, \emptyset, 10, 1, 01, \emptyset)$$

Using the multiplexed codebook, the generated bitstream is

$$000\ 100\ 100\ 110\ 100\ 001\ 011\ 111$$
Multiplexed Codes

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 $\mu_{i,i'}$

- Code construction
  - $\forall a_i. A$, a multiplexed code is designed for the conditional pdf $\mu_{i,i'}$
  - Equivalence classes indexed by $N_{i'}$

- For a stationary discrete first-order Markov source, the conditional entropy is reached iff
  \[ \forall (a_i, a_{i'}) \in A^2 \ |N_{i'}| = P(a_i \mid a_{i'}) \mid X \mid \]
Example

<table>
<thead>
<tr>
<th>$\mu_{i,i'}$</th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$a_3$</th>
<th>$S_{t-1}$</th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$a_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>a_1</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>000</td>
<td>001</td>
<td>001</td>
<td>001</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>001</td>
<td>001</td>
<td>001</td>
<td>001</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>010</td>
<td>010</td>
<td>010</td>
<td>010</td>
</tr>
<tr>
<td>a_2</td>
<td>8</td>
<td>2</td>
<td>4</td>
<td>011</td>
<td>011</td>
<td>011</td>
<td>011</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>4</td>
<td>2</td>
<td>101</td>
<td>101</td>
<td>101</td>
<td>101</td>
</tr>
<tr>
<td>a_3</td>
<td>8</td>
<td>4</td>
<td>2</td>
<td>110</td>
<td>110</td>
<td>110</td>
<td>110</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>4</td>
<td>2</td>
<td>111</td>
<td>111</td>
<td>111</td>
<td>111</td>
</tr>
</tbody>
</table>

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 \( k = \{ x \in \mathcal{X} / \exists a_i \in A, \forall a_i \in A, x \in \mathcal{C}_i \} \)
- In the example, \( |k| = 3 \)
- It could be 4

The kernel offers natural hard synchronization properties.
Error resilience analysis ($S_H$)

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

Transition probabilities of this model given by

$$P(\hat{S}_t = a_j, S_t = a_i | \hat{S}_{t-1} = a_j', S_{t-1} = a_i') = \frac{\mu_{i,j}}{N(a_j', X_t, Y_t, X_{t-1})} \sum_{R, Y_t, X_t} R, Y_t, X_t$$

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

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

$$\text{MSE}_K \to \infty = \sum_{(a_i, a_j) \in A^2} P(S_t = a_j, S_{t-1} = a_j; (a_j - a_i)^2)$$
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 $\rho = 0.5$
- 8 quantization cells
- $H_1(S_H) = 2.29$
- Binary symmetrical channel

- 1\textsuperscript{st}-order MUX-code:
  - $c = 5$
  - mdl = 2.35
Signal to Noise Ratio on a BSC

- $S_H$: Gauss-Markov source of correlation $\gamma = 0.5$
- 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

<table>
<thead>
<tr>
<th></th>
<th>Tab 1</th>
<th>Tab 2</th>
<th>Tab 3</th>
<th>Tab 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entropy</td>
<td>2.1543</td>
<td>3.3809</td>
<td>4.6972</td>
<td>5.4692</td>
</tr>
<tr>
<td>Mdl cavlc</td>
<td>2.1620</td>
<td>3.4062</td>
<td>4.7749</td>
<td>6 (FLC)</td>
</tr>
<tr>
<td>Mdl huffman</td>
<td>2.1608</td>
<td>3.4062</td>
<td>4.7338</td>
<td>5.5040</td>
</tr>
<tr>
<td>Mdl multiplexed codes</td>
<td><strong>2.1548</strong></td>
<td><strong>3.3871</strong></td>
<td><strong>4.6972</strong></td>
<td><strong>5.4693</strong></td>
</tr>
<tr>
<td>Symbols distribution</td>
<td>62%</td>
<td>25%</td>
<td>10%</td>
<td>1%</td>
</tr>
</tbody>
</table>

- 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