Turbo Source Coding

Laurent Schmalen and Peter Vary

FlexCode Public Seminar
June 16, 2008
Outline

• FlexCode in a nutshell
• Entropy coding
• Turbo coding and decoding
• Application of Turbo codes as source codes
• A joint-source channel coding scheme with iterative decoding for compression
• Possible Application to FlexCode
Who?

Ericsson

Nokia

KTH

RWTH Aachen

Orange (France Telecom)
The Problem

- Heterogeneity of networks increasing
- Networks inherently variable (mobile users)

But:
- Coders not designed for specific environment
- Coders inflexible (codebooks and FEC)
- Feedback channel underutilized
Adaptation and Coding

- adaptive coding
- AMR
- CELP with FEC
- rigid
- set of models, fixed quantizer
- PCM
- fixed model
- fixed quantizer
- flexible

IND - Institute of Communication Systems and Data Processing, RWTH Aachen University
Conventional Transmission

- **Transmitter**
  - Source
  - Parameter Extraction
  - Quantizer
  - Entropy Encoder
  - Channel Encoder
  - Sink

- **Receiver**
  - Signal Synthesis
  - Parameter Lookup
  - Entropy Decoder
  - Channel Decoder

\[ u \rightarrow \hat{u} \]
Prominent examples of lossless entropy coders
- Huffman coding
- Lempel-Ziv coding
- Arithmetic coding

Example: Huffman code
- AACABA \rightarrow 0 \ 0 \ 110 \ 0 \ 10 \ 0
- No bit pattern is prefix of another
- Unambiguous decoding

<table>
<thead>
<tr>
<th>Symbol</th>
<th>P(S)</th>
<th>Bit pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>0.3</td>
<td>10</td>
</tr>
<tr>
<td>C</td>
<td>0.15</td>
<td>110</td>
</tr>
<tr>
<td>D</td>
<td>0.05</td>
<td>111</td>
</tr>
</tbody>
</table>
• Usability of lossless source codes
• High sensitivity against transmission errors
  – Huffman code: synchronization loss
  – Arithmetic code: selection of wrong interval, complete decoding failure
Lossless Source Coding

- Usability of lossless source codes
- High sensitivity against transmission errors
  - Huffman code: synchronization loss
  - Arithmetic code: selection of wrong interval, complete decoding failure
- Very strong channel codes required
  - Error floor, i.e., seldom bit errors leading to decoding failures
• Usability of lossless source codes
• High sensitivity against transmission errors
  – Huffman code: synchronization loss
  – Arithmetic code: selection of wrong interval, complete decoding failure
• Very strong channel codes required
  – Error floor, i.e., seldom bit errors leading to decoding failures
• Iterative source-channel decoding schemes for Entropy Codes
  – High complexity
  – Difficult to apply to arithmetic coding [Guionnet 04]
• Wanted:
  – A flexible compression scheme (entropy coder) which
    • Has similar performance as known compression schemes
    • Is robust against transmission errors
    • Can instantaneously adapt to varying channel conditions by
      exchanging compression ratio against error robustness

  – Analogy between channel codes and source codes
    • A good channel code is often also a good source code
    • Use of LDPC codes for compression [Caire 03]

  – Can Turbo codes be used for compression?
Turbo Codes, Concept

- Bit mapping
- Turbo Encoder
- Modulation
- Physical Channel
- Demodulation
- Turbo Decoder
- Parameter Lookup

**Concatenated Encoding**

**Parallel scheme**

[Berrou 93]

**Iterative Turbo Decoding**

advantages by iterative feedback!
Turbo Codes, Concept

**Concatenated Encoding**
- Parallel scheme
  - [Berrou 93]
- Serial scheme
  - [Benedetto 98]

**Iterative Turbo Decoding**

Advantages by iterative feedback!
Turbo Coding & Decoding

- Turbo Code Encoder and Decoder [Berrou 93]

\[ \pi \]

denotes an interleaver

is an (almost) arbitrary channel encoder
• Turbo Code Encoder and Decoder [Berrou 93]

Encoder 1

Encoder 2

Transmission Channel

Decoder 1

Decoder 2

\( \pi \) denotes an interleaver

Encoder \( x \) is an (almost) arbitrary channel encoder
• Turbo Decoding (1st iteration, 1st step)
Turbo Coding & Decoding

- Turbo Decoding (1st iteration, 1st step)

Transmission over AWGN channel

- Data symbols
- Noisy symbols

Additive Gaussian noise

\[ \tilde{x}_i \]

\[ \tilde{y}_i^{(1)} \]

\[ \tilde{y}_i^{(2)} \]
• Turbo Decoding (1st iteration, 1st step)
• Turbo Decoding (1st iteration, 1st step)
• Turbo Decoding (1st iteration, 2nd step)
Turbo Coding & Decoding

• Turbo Decoding (1st iteration, 2nd step)
• Turbo Decoding (2nd iteration, 1st step)
• Turbo Decoding (2nd iteration, 1st step)
• Turbo Decoding (2nd iteration, 1st step)
• Turbo Decoding (2nd iteration, 2nd step)
• Turbo Decoding (2nd iteration, 2nd step)
• Turbo Decoding (2nd iteration, 2nd step)
• Turbo Decoding ($n$th iteration, 1st step)
Turbo Decoding ($n$'th iteration, 1st step)
• Turbo Decoding ($n$'th iteration, 2nd step)
• Turbo Decoding ($n$'th iteration, 2nd step)
- Block interleaver vs. random interleaver
  - Example: Propagation of a single information
    - Block: (50 x 50)
Turbo Principle - Interleaver Design

- Block interleaver vs. random interleaver
  - Example: Propagation of a single information
  - Block: (50 x 50)
  - Random:
  - Better information distribution with random interleavers
  - Careful interleaver design required
Can Turbo codes be used for entropy coding?

Yes!

- Turbo Codes for compressing binary memoryless sources [Garcia-Frias 02]
- Only transmitting a fraction of the output bits such that the overall coding rate > 1
- Decoder has to take into consideration the statistics of the source (unequal distribution of bits)
- Source statistics can be estimated at the decoder
Transmission with Turbo Source Coding

- **Transmitter**
  - Source
  - Parameter Extraction
    - $\tilde{u}$
  - Quantizer
  - Bit mapping
  - Turbo Encoder

- **Receiver**
  - Signal Synthesis
  - $\hat{u}$
  - Parameter Lookup
  - Turbo Decoder
Transmission with Turbo Source Coding

**Transmitter**

- **Source**
  - Parameter Extraction
  - Quantizer
    - \( u \)
  - Bit mapping
  - Turbo Encoder

**Receiver**

- Signal Synthesis
  - \( \hat{u} \)
  - Parameter Lookup
  - Turbo Decoder
Transmission with Turbo Source Coding

• Transmitter

- Quantizer
- Parameter Extraction
- Source

• Bit mapping

Mapping of quantizer reproduction levels to bit patterns:

- $0 \rightarrow 000$
- $1 \rightarrow 001$
- $2 \rightarrow 010$
- ...
- $7 \rightarrow 111$

• Turbo Encoder

- Sink

• Receiver

- Turbo Decoder
- Parameter Lookup

$\hat{u}$
Turbo Source Coding

- Turbo coder, channel coding rate 1/3 (1 bit → 3 bit)
• Turbo coder, channel coding rate 1/3 (1 bit → 3 bit)

\[ y = (x_1, y_1^{(1)}, y_1^{(2)}, x_2, y_2^{(1)}, y_2^{(2)}, x_3, y_3^{(1)}, y_3^{(2)}, \ldots) \]
Turbo Source Coding

- Turbo coder, channel coding rate 1/3 (1 bit $\rightarrow$ 3 bit)

- Before puncturing:
  \[ y = (x_1, y_1^{(1)}, y_1^{(2)}, x_2, y_2^{(1)}, y_2^{(2)}, x_3, y_3^{(1)}, y_3^{(2)}, \ldots) \]

- After puncturing (compression ratio 0.5, 1 bit $\rightarrow$ 0.5 bit)
  \[ y_p = (x_1, y_3^{(1)}, y_5^{(2)}, x_7, y_9^{(1)}, y_{11}^{(2)}, \ldots) \]
Turbo Source Coding

- Puncturing unit corresponds to binary erasure channel
- Adapt puncturing w.r.t. source statistics
- Theoretical minimum rate: source entropy
- Realization of puncturing
  - Regular puncturing
  - Pseudo-random puncturing
- Puncturing has to be known at the receiver
- Adaptively increase number of transmitted bits with increasing channel noise
Turbo Source Coding

- Simulation results for a binary source [Garcia-Frias 02]
- Comparison with standard Unix compression tools `compress`, `gzip`, and `bzip2`

<table>
<thead>
<tr>
<th>Source entropy $H(X)$</th>
<th>Achieved compression ratios</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Turbo</td>
</tr>
<tr>
<td>0.0808</td>
<td>0.16</td>
</tr>
<tr>
<td>0.2864</td>
<td>0.38</td>
</tr>
<tr>
<td>0.4690</td>
<td>0.58</td>
</tr>
<tr>
<td>0.6098</td>
<td>0.75</td>
</tr>
<tr>
<td>0.7219</td>
<td>0.87</td>
</tr>
</tbody>
</table>
Turbo Source Coding

- **Advantages:**
  - Robustness against transmission errors
  - If the channel quality drops, puncturing can be changed in order to transmit more parity bits
  - High flexibility by adapting puncturing on the fly

- **Disadvantages:**
  - Higher computational costs than conventional entropy coding schemes
  - Lossless compression is not guaranteed, Turbo decoder might not decode every bit correctly
  - Difficult to adapt to varying parameter statistics for the use in speech and audio codecs
Lossless Turbo Source Coding

- Lossless Turbo source coding [Hagenauer 04]
- Adapt puncturing such that lossless decoding is possible
- Analysis-by-Synthesis encoder:
  - change puncturing and test if decodable without error
  - Small amount of side information required
Non-Binary Sources

• Turbo Source Coding only for binary sources
• Feasible only if bit pattern after source coding has low entropy $H(X) < 1$
• Extension towards non-binary sources
  – Utilization of non-binary Turbo codes [Zhao 02]
  – Utilization of special binary LDPC [Zhong 05]
  – Utilization of non-binary LDPC codes [Potluri 07]
• Joint source-channel coding approach

„However, any redundancy in the source will usually help if it is utilized at the receiving point. [...] redundancy will help combat noise.“, Shannon 1948
Iterative Source-Channel Decoding

- Approach: Enhancement of the robustness of transmission of variable length codes
- Iterative Source-Channel Decoding (ISCD) of variable length codes (VLC) [Guyader 01]
  - Combats adverse effects of channel noise
  - Exploits the structure and redundancy of variable length codes
  - Achieve near-capacity system performance
  - Works considerably well for Huffman codes
  - Difficult to adapt to arithmetic codes
  - Very high computational complexity
  - Limited flexibility!
Iterative Source-Channel Decoding

- Approach: Enhancement of the robustness of transmission of variable length codes
- Iterative Source-Channel Decoding (ISCD) of variable length codes (VLC) [Guyader 01]
  - Combats adverse effects of channel noise
  - Exploits the structure and redundancy of variable length codes
  - Achieve near-capacity system performance
  - Works considerably well for Huffman codes
  - Difficult to adapt to arithmetic codes
  - Very high computational complexity
  - Limited flexibility!
Iterative Source-Channel Decoding

- Iterative Source-Channel Decoding (ISCD) [Adrat 01] for fixed-length codes (FLC)
  - Combat adverse effects of channel noise
  - Exploit residual source redundancy
  - Achieve near-capacity overall system performance [Clevorn 06]

- Can also be used effectively for compression [Thobaben 08]
  - Leave all redundancy in the source symbols
  - Utilization of redundant bit mappings
  - Puncture output of convolutional code in order to obtain compression
Turbo Codes, Concept

Concatenated Encoding
- Parallel scheme
  - [Berrou 93]
- Serial scheme
  - [Benedetto 98]

Iterative Turbo Decoding

Advantages by iterative feedback!
Turbo Codes, Concept

- Concatenated Encoding
  - serial scheme
  - [Benedetto 98]

- Iterative Turbo Decoding
  - advantages by iterative feedback!
Transmission with ISCD

- **Transmitter**
  - Source
  - Parameter Extraction
  - Quantizer
  - Redundant Bit mapping
  - Channel Encoder

- **Receiver**
  - Sink
  - Signal Synthesis
  - Channel Decoder
  - Soft Decision Source Decoder
  - \( \hat{u} \)
Transmission with ISCD

- **Transmitter**
  - Source
  - Parameter Extraction
  - Quantizer
  - Redundant Bit Mapping
  - Channel Encoder

- **Receiver**
  - Soft Decision Source Decoder
  - Channel Decoder
  - Sink

\[ \hat{u} \]
Transmission with ISCD

- **Transmitter**
  - Quantizer
  - Parameter Extraction
  - Source
  - Redundant Bit mapping
  - Mapping of quantizer reproduction levels to bit patterns. Additional redundancy, e.g., by parity check bit
    0 → 000 0
    1 → 001 1
    2 → 010 1
    ...
    7 → 111 1
  - Channel Encoder

- **Receiver**
  - Channel Decoder
  - Source Decoder
  - Soft Decision
  - Sink
  - Signal Synthesis
  - \( \hat{u} \)
Soft Decision Source Decoding

Exploitation of residual redundancy for quality improvement

1D a priori knowledge (parameter distribution)

Calculate extrinsic feedback information using source statistics and bit mapping redundancy [Adrat 01]
Exploitation of residual redundancy for quality improvement

Calculate extrinsic feedback information using source statistics of order 0 and 1 and bit mapping redundancy [Adrat 01]
ISCD Source Compression

- Simulation example
- System components
  - Scalar quantization with $Q$ levels
  - Single parity check code index assignment
  - R > 1 convolutional code, random puncturing [Thobaben 07]

$K(\log_2(Q) + 1) \div N$ bits

$K$ parameters

Bit mapping

$N_C < N$ bits

Puncturing
Iterative Source-Channel Decoding

- Simulation results for simple experiment
  - Gauss-Markov source (AR(1) process)
  - Lloyd-Max Quantization with \( Q = 16 \) levels
  - 25 decoding iterations
  - Unoptimized standard system components!

| Source correlation | Entropy \( H(U_t | U_{t-1}) / 4 \) | Achieved compression ratios |
|--------------------|----------------------------------|----------------------------|
| \( \rho \)         |                                  | ISCD | compress | gzip  | bzip2 |
| 0.8                | 0.76                             | 0.83 | 0.99     | 0.86  | 0.82  |
| 0.85               | 0.72                             | 0.78 | 0.94     | 0.83  | 0.78  |
| 0.9                | 0.66                             | 0.74 | 0.87     | 0.80  | 0.72  |
| 0.95               | 0.55                             | 0.62 | 0.73     | 0.71  | 0.62  |
Possible Application in FlexCode

- Entropy coding for constrained entropy quantization

- Utilization of entropy coding, e.g., arithmetic coding
- Can be replaced by the presented compression scheme
Possible Application in FlexCode

- Entropy coding for constrained entropy quantization

- Flexible adaptation to varying channel conditions by puncturing adaptation
Conclusions

• Channel codes can be used for compression
• Turbo codes for compressing binary sources
• Iterative Source-Channel Decoding for fixed length codes
• Joint Source-Channel Coding with Iterative Decoding for Source Compression
• Promising results already with unoptimized system components
### References

**[Guionnet 04]**  

**[Caire 03]**  

**[Berrou 93]**  
C. Berrou, A. Glavieux, P. Thitimajshima, „Near Shannon Limit Error-Correcting Coding and Decoding: Turbo Codes“, *International Conference on Communications*, Geneve, Switzerland, Mai, 1993

**[Benedetto 98]**  

**[Garcia-Frias 02]**  
J. Garcia-Frias, Y. Zhao, „Compression of Binary Memoryless Sources Using Punctured Turbo Codes“, *IEEE Comm. Letters*, vol. 6, no. 9, September 2002

**[Hagenauer 04]**  

**[Zhao 02]**  
Y. Zhao, J. Garcia-Frias, „Data Compression of Correlated Non-Binary Sources Using Punctured Turbo Codes“, *IEEE Data Compression Conference (DCC)*, 2002

**[Zhong 05]**  

**[Potluri 07]**  
M. Potluri, S. Chilumuru, S. Kambhampati, K. R. Namuduri, „Distributed Source Coding using non-binary LDPC codes for sensor network applications“, *Canadian Workshop on Information Theory*, June 2007

**[Guyader 01]**  

**[Adrat 01]**  
<table>
<thead>
<tr>
<th>Reference</th>
<th>Description</th>
</tr>
</thead>
</table>
Turbo Source Coding

Laurent Schmalen and Peter Vary

http://www.flexcode.eu