Error-resilient source codes
and joint source/channel codes

Hervé Jégou

Université de Rennes I /IRISA

November 2005
Outline

- Framework and related work
- Multiplexed codes
- Soft decoding of variable length codes
- Variable length re-writing systems
- Conclusion and perspectives
Framework

- Transmission of a signal over a noisy channel

Objective: the best reconstruction w.r.t. the rate → with a reasonable complexity
Transmission setup

- Practical source coding scheme:

![Source coder diagram]

- Transform
- Quantization
- Entropy coding

00110101...
Transmission setup

- Practical source coding scheme:

- In the traditional setup, source coding does not care about channel errors
Transmission setup

Practical source coding scheme:

→ Error correcting codes in a separated setup (Shannon separation theorem)
Joint source/channel: motivation

- Channel coding techniques are effective against channel noise
  - But channel characteristics must be known at the time of the encoding and may vary in time
    → residual bit error rate
  - Also, the separation theorem does not take into account some important practical constraints: bounded delay and computational power
→ Joint source/channel schemes have gained attention in recent years
Related work

- Unequal error protection
  - several sources
  - overall rate/distortion optimization

Source and channel coding are performed jointly
Sensitivity of entropy coding

In this thesis: focus on the entropy coding step
  - intrinsically error-resilient source codes
  - joint entropy/channel codes

Lossless source coding: discrete sources only
Outline

- Framework and related work
- Multiplexed codes
- Soft decoding of variable length codes
- Variable length re-writing systems
- Conclusion and perspectives
Motivation

To find entropy codes that are intrinsically protected against the desynchronization phenomenon

- fixed length codes $\rightarrow$ no compression
- Tunstall Codes (1967) and self-synchronizing VLCs [Ferguson & al. 84]
  $\rightarrow$ synchronization for the Levenshtein distance only

For one source, the output is of variable length
Multiplexed codes

- $S_H$: high priority source on a finite alphabet $\mathcal{A}$
  $\mathcal{A} = \{a_1, \ldots, a_i, \ldots\}$
- $S_L$: binary source of lower priority
- The key idea: an entropy code protecting source $S_H$ against error propagation
- Unequal error-resilience at the entropy coder level
Set of codewords $\mathcal{X}$ such that $|\mathcal{X}| \geq |\mathcal{A}|$

Let us assume that $|\mathcal{A}| = 5$ and $|\mathcal{X}|$ is chosen as $2^3$
Principle

<table>
<thead>
<tr>
<th>$a_i$</th>
<th>$x \in \mathcal{X}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td>000</td>
</tr>
<tr>
<td></td>
<td>001</td>
</tr>
<tr>
<td></td>
<td>010</td>
</tr>
<tr>
<td>$a_2$</td>
<td>011</td>
</tr>
<tr>
<td></td>
<td>100</td>
</tr>
<tr>
<td>$a_3$</td>
<td>101</td>
</tr>
<tr>
<td>$a_4$</td>
<td>110</td>
</tr>
<tr>
<td>$a_5$</td>
<td>111</td>
</tr>
</tbody>
</table>

- $\mathcal{X}$ is partitioned into $|\mathcal{A}|$ subsets $C_i$
- $C_i = \text{equivalence classes for } S_H: S_t = a_i \iff X_t \in C_i$
Principle

<table>
<thead>
<tr>
<th>$a_i$</th>
<th>$\mathbb{P}(a_i)$</th>
<th>$x \in \mathcal{X}$</th>
<th>index $q_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td>0.4</td>
<td>000</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>001</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>010</td>
<td>2</td>
</tr>
<tr>
<td>$a_2$</td>
<td>0.2</td>
<td>011</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>1</td>
</tr>
<tr>
<td>$a_3$</td>
<td>0.2</td>
<td>101</td>
<td>0</td>
</tr>
<tr>
<td>$a_4$</td>
<td>0.1</td>
<td>110</td>
<td>0</td>
</tr>
<tr>
<td>$a_5$</td>
<td>0.1</td>
<td>111</td>
<td>0</td>
</tr>
</tbody>
</table>

- Remaining storage capacity available with index $q_i$
- The cardinals $|C_i|$ should be proportional to $\mathbb{P}(a_i)$
Multiplexed codes

- The key idea: a redundant FLC designed for $S_H$
  - $S_H$ inherits FLC properties, i.e.
    - random access to data
    - strict-sense synchronization
- Redundancy used to describe $S_L$
- Compression efficiency (expected description length):

  $$\text{EDL}(S_H) = - \sum_{a_i \in A} P(a_i) \log_2 \frac{|C_i|}{|\mathcal{X}|}$$

- Entropy is reached if $|C_i|$ are such that $|C_i| = P(a_i)|\mathcal{X}|$
Multiplexing algorithm

- How to exploit the redundancy of the FLC? (How to choose the sequence representing $s_H$?)

<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></td>
<td></td>
<td></td>
<td>010</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 $s_H$: $3 \times 1 \times 1 \times 2 \times 2 \times 1 \times 1 \times 3 \times 2 = 36$

- Multiplexing capacity: $\log_2 36 \approx 5.17$ bits
Conversion of $S_L$

- 5 bits are taken from the binary source $S_L$
- These bits are seen as the binary representation of an integer $\gamma$
- $\gamma$ is transformed into a sequence of states $(q_t)_t$ using Euclidean divisions as
  \[
  \gamma = q_1 + n_1(q_2 + n_2(\ldots + n_t(q_{t+1} + \ldots)\ldots)\ldots),
  \]
  where $n_t$ is the realization of the cardinal of the equivalence class associated to the symbol $S_t$
- Problem: computational cost of the corresponding long integer operation has complexity $O(L(S_H)^2)$
Hierarchical algorithm

- Dichotomic processing of $\gamma$

\[ n_1^1 = 3 \quad n_2^3 = 1 \quad n_3^3 = 1 \quad n_4^4 = 2 \quad n_5^5 = 1 \quad n_6^6 = 1 \quad n_7^7 = 3 \quad n_8^8 = 2 \]
Hierarchical algorithm

- Dichotomic processing of $\gamma$

\[ n_1^1 = 3 \quad n_2^2 = 1 \quad n_3^3 = 1 \quad n_4^4 = 2 \]
\[ n_5^5 = 1 \quad n_6^6 = 1 \quad n_7^7 = 3 \quad n_8^8 = 2 \]
Hierarchical algorithm

Dichotomic processing of $\gamma$

\[
\begin{align*}
n_1 &= 3 & n_2 &= 1 & n_3 &= 1 & n_4 &= 2 & n_5 &= 1 & n_6 &= 1 & n_7 &= 3 & n_8 &= 2 \\
n_1 &= 3 & n_2 &= 1 & n_3 &= 1 & n_4 &= 2 & n_5 &= 1 & n_6 &= 1 & n_7 &= 3 & n_8 &= 2 \\
n_1 &= 6 & n_4 &= 2 & n_5 &= 1 & n_6 &= 1 & n_7 &= 3 & n_8 &= 2
\end{align*}
\]

$\Lambda = n_1^8 = 36$  \hspace{1cm}  5 bits of $s_L$  \hspace{1cm}  $\gamma = q_1^8 = 26$
Hierarchical algorithm

- Dichotomic processing of $\gamma$

\[
\begin{align*}
n_1 &= 3 \\
n_2 &= 1 \\
n_3 &= 1 \\
n_4 &= 2 \\
n_5 &= 1 \\
n_6 &= 1 \\
n_7 &= 3 \\
n_8 &= 2 \\
\end{align*}
\]

$\Lambda = n_1^8 = 36$

$\gamma = q_1^8 = 26$

$q_1^4 = 2$

$q_5^8 = 4$

$\text{mod}(26,6)$

$26/6$
Hierarchical algorithm

Dichotomic processing of $\gamma$

$n_1^1 = 3 \quad n_2^1 = 1 \quad n_3^3 = 1 \quad n_4^4 = 2 \quad n_5^5 = 1 \quad n_6^6 = 1 \quad n_7^7 = 3 \quad n_8^8 = 2$

$n_1^2 = 3 \quad n_3^2 = 2 \quad n_5^5 = 1 \quad n_8^8 = 6$

$n_1^4 = 6 \quad n_5^8 = 6 \quad n_8^8 = 6$

$\Lambda = n_1^1 = 36$

$\mod(26,6) \quad \gamma = q_1^8 = 26 \quad 26/6$

$q_1^2 = 2 \quad q_3^4 = 0 \quad q_5^6 = 0 \quad q_7^8 = 4$

$q_1^4 = 2 \quad q_5^8 = 4$
Hierarchical algorithm

- Dichotomic processing of $\gamma$

\[
\begin{align*}
 n_1^1 &= 3 & n_2^2 &= 1 & n_3^3 &= 1 & n_4^4 &= 2 & n_5^5 &= 1 & n_6^6 &= 1 & n_7^7 &= 3 & n_8^8 &= 2 \\
 n_1^1 &= 3 & n_3^3 &= 2 & n_5^5 &= 1 & n_7^7 &= 6 & n_8^8 &= 6 \\
 n_1^4 &= 6 & n_5^8 &= 6 \\ 
\Lambda &= n_1^1 = 36
\end{align*}
\]

- $\mod(26,6) \gamma = q_1^8 = 26$\hspace{1cm} 26/6

\[
\begin{align*}
 q_1^2 &= 2 & q_3^4 &= 0 & q_5^6 &= 0 & q_7^8 &= 4 \\
 q_1^1 &= 2 & q_3^3 &= 0 & q_4^4 &= 0 & q_6^6 &= 0 & q_8^8 &= 1 \\
 q_2^2 &= 0 & q_4^3 &= 0 & q_5^6 &= 0 & q_7^7 &= 1 \\
\end{align*}
\]

- The complexity is in $O(L(S_H)^r)$, where $r$ is close to 1
Example
Example

\[ s_H \]

\[ a_1 \quad a_4 \quad a_5 \quad a_2 \quad a_3 \quad a_3 \quad a_1 \quad a_2 \]

\[ C_1 \quad C_4 \quad C_5 \quad C_2 \quad C_3 \quad C_3 \quad C_1 \quad C_2 \]

\[ 000 \quad 001 \quad 110 \quad 111 \quad 011 \quad 010 \quad 101 \quad 101 \quad 000 \quad 001 \quad 011 \quad 010 \]

\[ (n_i)_i \]

\[ \gamma = 26 \]

\[ s_L \]

\[ 1 \quad 1 \quad 0 \quad 1 \quad 0 \]
Example

\[ s_H \]

\[ a_1 \quad a_4 \quad a_5 \quad a_2 \quad a_3 \quad a_3 \quad a_1 \quad a_2 \]

\[ C_1 \quad C_4 \quad C_5 \quad C_2 \quad C_3 \quad C_3 \quad C_1 \quad C_2 \]

\[ (q_l) \]

\[ 2 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \]

\[ \gamma = 26 \]

\[ s_L \]

\[ 1 \quad 1 \quad 0 \quad 1 \quad 0 \]

Error-resilient source codes and joint source/channel codes – p.15/50
Example
Simulation results on a BSC

- $S_H$ and $S_L$: identical sources of 5 symbols
- $S_L$ pre-encoded with a VLC
- $S$: same source as $S_H$ and $S_L$
A single source $S$ arbitrarily separated in two parts $S_H$ and $S_L$
A simple image coder

Bit error rate = 0.0005

Fixed length codes

25.50 dB
bpp=6.187

Huffman codes

24.78 dB
bpp=1.713

Multiplexed codes

30.71 dB
bpp=1.712
A simple image coder

Bit error rate = 0.005

<table>
<thead>
<tr>
<th>Fixed length codes</th>
<th>Huffman codes</th>
<th>Multiplexed codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>16.79 dB</td>
<td>15.59 dB</td>
<td>24.82 dB</td>
</tr>
<tr>
<td>bpp=6.187</td>
<td>bpp=1.713</td>
<td>bpp=1.712</td>
</tr>
</tbody>
</table>
A simple image coder

Bit error rate = 0.05

Fixed length codes

Huffman codes

Multiplexed codes

13.99 dB
bpp=6.187

12.64 dB
bpp=1.713

19.63 dB
bpp=1.712
First-order multiplexed codes

- Motivation: to exploit the conditional probabilities
- Code construction
  - $\forall a_i \in \mathcal{A}$, a memoryless multiplexed code is designed for the conditional probability mass function $\mathbb{P}(S_t = a_i | S_{t-1} = a_{i'})$
  - Equivalence classes are indexed by $C_i^{i'}$
- For a stationary discrete first-order Markov source, the entropy rate is reached iff

\[
\forall (a_i, a_{i'} \in \mathcal{A}^2, |C_i^{i'}| = \mathbb{P}(S_t = a_i | S_{t-1} = a_{i'})|\mathcal{X}|
\]
Example

\[
P(a_i | a_{i'}) \begin{array}{c|ccc} 
 & a_1 & a_2 & a_3 \\
\hline 
a_1 & \frac{3}{4} & \frac{1}{4} & \frac{1}{4} \\
a_2 & \frac{1}{8} & \frac{5}{8} & \frac{1}{4} \\
a_3 & \frac{1}{8} & \frac{1}{8} & \frac{1}{2} \\
\end{array}
\]

→ entropy rate is reached
The variable $Z_t = (S_t, \hat{S}_t)$ forms a Markov chain for which we can compute the transition probabilities.
Error-resilience analysis

- The steady state gives us $\mathbb{P}(S_t = a_i, \hat{S}_t = a_{iv})$
  $\Rightarrow$ asymptotic values of the SER and SNR

- The expressions can then be optimized by considering different choices for the index assignment

- The optimization is carried out using
  - the binary switching algorithm
    [Zeger & Gersho 90]
  - simulated annealing optimization techniques
    [Farvardin 91]
  - a specific approach
Symbol error rate on a BSC

Error-resilient source codes and joint source/channel codes – p.22/50
Multiplexed codes: other results

- Several code construction methods → approaching the entropy of the source
- Efficient methods to convert $S_L$
  - binary multiplexed codes
  - constrained partitions
- Soft decoding algorithms
- Strict synchronization of first-order multiplexed codes enforced by periodic use of memoryless multiplexed codes
Outline

- Framework and related work
- Multiplexed codes
- Soft decoding of variable length codes
- Variable length re-writing systems
- Conclusion and perspectives
Motivation

Objective: to exploit the suboptimality of the entropy coding on the decoder side
→ Bayesian estimation at the decoder [Balakirsky 97]
Motivation

Objective: to exploit the suboptimality of the entropy coding on the decoder side
→ Bayesian estimation at the decoder [Balakirsky 97]

possibly coupled with a convolutional code
→ Turbo-VLC [Hagenauer 00] [Guyader & al 01]
Decoding trellises

- The bit-level trellis [Balakirsky 97]
- The states are the internal nodes \((n_k)
- Termination constraint: the trellis terminate in the root node
- Computing cost: \(O(|A| \times L)\)
- VLC tables can be simplified [Mohammad-Khani, Kieffer & al. 05]
Decoding trellises

- The bit/symbol trellis [Park & Miller 98] [Hagenauer 00]
- Decoding with symbol length constraint
- The states are defined as \((n_k, t_k)\) so that the memory of the symbol clock is preserved
- Prohibitive computing cost: \(O(|A| \times L^2)\) → suboptimal sequential decoding techniques
State aggregation

- Aggregated trellis
  - The states are defined as \((n_k, t_k \mod T)\)
  - Decoding with symbol length constraint → partial information on the length is preserved

- Computing cost: \(O(|A| \times T \times L)\) → optimal decoding w.r.t. the aggregated model
Impact of the aggregation

- $T = 1$ bit/level trellis
- $T = L(S) \iff$ bit/symbol trellis
- $1 < T < L(S) \Rightarrow$ trade-off between complexity and estimation accuracy

$T = 100$

$T = 20$

$T = 10$
Decoding performance

Symbol error rate vs $E_b/N_0$ for different values of $T$. The graph shows the performance of hard decoding for various values of the parameter $T$. The curves are labeled with $T=1$, $T=2$, $T=3$, $T=5$, and $T=10$. The symbol error rate decreases as $E_b/N_0$ increases, indicating improved decoding performance with higher signal-to-noise ratio.
Decoding performance

![Graph showing symbol error rate vs. Eb/N0 for different values of T and trellis type.](image-url)
Decoding performance

Symbol error rate vs. $E_b/N_0$ for different trellis complexities (T=1, T=2, T=3, T=5, T=10) and hard-decision decoding. The graph shows the improvement in symbol error rate with increasing $E_b/N_0$ for each trellis complexity.
Decoding performance

![Graph showing decoding performance with symbol error rate vs. $E_b/N_0$ for different values of $T$.](image-url)
Decoding performance

![Graph showing decoding performance](image-url)

- Symbol error rate
- $E_b/N_0$
Decoding performance

![Graph showing decoding performance with bit-level trellis and various error rates.](image)
Analysis

- Starting point: gain/loss analysis [Swaszek & al. 95]
- Extension of this analysis to measure
  - the amount of information conveyed by the termination constraint and that is likely to be exploited
  → criterion for finding the best codes
- The best codes are the ones that have *poor* resynchronization properties
- For $T$ high enough, the trellis aggregation has a negligible impact on the information conveyed by the termination constraint
Outline

- Framework and related work
- Multiplexed codes
- Soft decoding of variable length codes
- Variable length re-writing systems
- Conclusion and perspectives
Motivation

- To attain the capacity of a BSC, the binary input of this channel should be uniform.

- However,
  - 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

- We addressed this problem from a source soft decoding perspective.
Variable length re-writing systems

- VLRS: variable length re-writing system

- VLRS ≡ Set of production rules of the form

\[ r : a \bar{l} \rightarrow \bar{b}, \]

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

- Note: these rules must satisfy some additional constraints
Variable length re-writing systems

Examples of codes and tree representations:

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

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

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

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

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

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

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

\[
\begin{array}{ll}
  r_{1,1} : & a_1 a_1 a_1 a_1 a_1 1 \\
  r_{1,2} : & a_1 a_1 a_1 a_1 0 \\
\end{array}
\]
Encoding principle

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

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

\[ r_{1,1} : a_1 \ a_1 \ a_1 \ a_1 \ a_1 \ a_1 \ 1 \]
\[ r_{1,2} : a_1 \ a_1 \ a_1 \ a_1 \ a_1 \ 0 \]
\[ r_{1,1} : a_1 \ a_1 \ a_1 \ 1 \ 0 \]
Encoding principle

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

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

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

<table>
<thead>
<tr>
<th>$r_{1,1}$</th>
<th>$a_1 1$</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_{1,2}$</td>
<td>$a_1 0$</td>
<td>10</td>
</tr>
<tr>
<td>$r_{2,1}$</td>
<td>$a_2$</td>
<td>110</td>
</tr>
<tr>
<td>$r_{3,1}$</td>
<td>$a_3$</td>
<td>111</td>
</tr>
<tr>
<td>$r_{1,1}$</td>
<td>$a_1 a_1 a_1 a_1 a_1 a_1 1$</td>
<td></td>
</tr>
<tr>
<td>$r_{1,2}$</td>
<td>$a_1 a_1 a_1 a_1 a_1 0$</td>
<td></td>
</tr>
<tr>
<td>$r_{1,1}$</td>
<td>$a_1 a_1 a_1 1 0$</td>
<td></td>
</tr>
<tr>
<td>$r_{1,2}$</td>
<td>$a_1 a_1 0 0$</td>
<td></td>
</tr>
<tr>
<td>$r_{1,1}$</td>
<td>$a_1 1 0 0$</td>
<td></td>
</tr>
</tbody>
</table>
Encoding principle

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

<table>
<thead>
<tr>
<th>$r_{1,1}$</th>
<th>$a_11$</th>
<th>$0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_{1,2}$</td>
<td>$a_10$</td>
<td>$10$</td>
</tr>
<tr>
<td>$r_{2,1}$</td>
<td>$a_2$</td>
<td>$110$</td>
</tr>
<tr>
<td>$r_{3,1}$</td>
<td>$a_3$</td>
<td>$111$</td>
</tr>
</tbody>
</table>

emitted sequence: 000
Finite state machines

- Encoding and decoding FSMs are constructed from the set of rules, generating trellises.
- Example: decoding FSM and corresponding trellis.
Compression efficiency

- Compression efficiency $\equiv$ expected number of bits produced by a production rule

- $(R_L(S), ..., R_t, R_{t-1})$ forms a Markov chain
  $\Rightarrow$ the asymptotic compression efficiency is deduced from the steady state $\pi = P(R_t)$

- Example: for previous codes and $P(a_i) = \{0.7, 0.2, 0.1\}$
  $\text{EDL(VLRS)} = 1.188 \text{ bit} < \text{EDL(Huffman)} = 1.3 \text{ bit}$
Mirrored Construction

- Motivation: to design a VLRS leading to a uniform distribution of output bits
- Method: construction from a VLC (same EDL):

```
  0  1  1
a_1 a_2 a_3

  0  0  1
a_3 a_2 a_1

  0  0  1
a_1 0 a_3 1 a_2 1
```
Soft decoding performance

\[ \text{Code}\ \{00, 11, 010, 101, 0110\}, \]
\[ \mathbb{P}(a_i) = \{0.4, 0.2, 0.2, 0.1, 0.1\} \]

\[ \Rightarrow \mathbb{P}(0) = 0.6 \]

- Reversible VLC vs mirror VLRS
- Backward soft decoding
- Balakirsky-like state model
- Viterbi decoding
Soft decoding performance

Code \{00, 11, 010, 101, 0110\},
\(\mathbb{P}(a_i) = \{0.9, 0.025, \cdots\}\)

\(\Rightarrow \mathbb{P}(0) = 0.917\)

- Reversible VLC
- vs mirror VLRS
- Backward soft decoding
- Balakirsky-like state model
- Viterbi decoding
Outline

- Framework and related work
- Multiplexed codes
- Soft decoding of variable length codes
- Variable length re-writing systems
- Conclusion and perspectives
Can we construct an optimal entropy code such that the source is protected against the desynchronization phenomenon?

For one source among two, YES

- multiplexed codes
Conclusion (2)

Can we reduce the computational cost of the soft decoding with length constraint?

For jointly typical source/channel realizations, YES

- aggregated state model
- validated by simulations
Conclusion (3)

Should we seek VLCs with good strict-sense synchronization properties?

I would say YES if hard decoding is processed at the decoder.
Conclusion (3)

Should we seek VLCs with good strict-sense synchronization properties?

I would say YES if hard decoding is processed at the decoder

I would say NO if soft decoding with length constraint is used instead
Perspectives

- State aggregation for quasi-arithmetic codes
- VLRS
  - new design methods are needed
  - to be used in a iterative structure
  - pure joint/source channel codes, in the spirit of variable length error correcting codes?
- To be used in real coders
Questions

?