Acknowledgements

People:

- Andreas Winkelbauer
- Clemens Novak
- Stefan Schwandter

Funding:

SISE - Information Networks (FWF Grant S10606)
Outline

1. Introduction

2. Soft channel encoding

3. Approximations

4. Applications

5. Conclusions and outlook
Outline

1. Introduction

2. Soft channel encoding

3. Approximations

4. Applications

5. Conclusions and outlook
Introduction: Basic Idea

Soft information processing
- popular in modern receivers
- idea: use soft information also for transmission

Soft-input soft-output (SISO) encoding
- extension of hard-input hard-output (HIHO) encoding
- data only known in terms of probabilities
- how to (efficiently) encode “noisy” data?

Applications
- physical layer network coding
- distributed (turbo) coding
- joint source-channel coding
Example: relay channel

- relay R assists S – D transmission
- decode-and-forward → distributed channel coding
- D can perform iterative (turbo) decoding

What if relay fails to decode?

- forwarding soft information might help
- how to transmit soft information?
Introduction: Motivation (2)

Soft information forwarding


Transmission of soft information

• LLR quantization ⇒ information bottleneck

• quantizer labels & symbol mapping ⇒ binary switching
1. Introduction

2. Soft channel encoding

3. Approximations

4. Applications

5. Conclusions and outlook
Hard Encoding/Decoding Revisited

Notation

- information bit sequence: \( u = (u_1 \ldots u_K)^T \in \{0, 1\}^K \)
- code bit sequence: \( c = (c_1 \ldots c_N)^T \in \{0, 1\}^N \)
- assume \( N \geq K \)

Linear binary channel code

- one-to-one mapping \( \phi \) between data bits \( u \) and code bits \( c \)
- encoding: \( c = \phi(u) \)
- codebook: \( C = \phi(\{0, 1\}^K) \)

Hard decoding:

- observed code bit sequence \( c' = c \oplus e = \phi(u) \oplus e \)
- decoder: mapping \( \psi \) such that \( \psi(c') \) is “close” to \( u \)
Soft Decoding Revisited

**Word-level**

- observations: code bit sequence probabilities $p_{\text{in}}(c')$
- enforce code constraint:

$$p'(c) = \begin{cases} \frac{p_{\text{in}}(c)}{\sum_{c' \in C} p_{\text{in}}(c')}, & c \in C \\ 0, & \text{else} \end{cases}$$

- info bit sequence probabilities: $p_{\text{out}}(u) = p'(\phi(u))$
- conceptually simple, computationally infeasible

**Bit-level**

- observed: code bit probabilities $p_{\text{in}}(c_n) = \sum_{c' \sim c_n} p_{\text{in}}(c')$
- desired: info bit probabilities $p_{\text{out}}(u_k) = \sum_{u' \sim u_k} p_{\text{out}}(\phi(u))$
- conceptually more difficult, computationally feasible
- example: equivalent to BCJR for convolutional codes
Soft Encoding: Basics

Word-level

• given: info bit sequence probabilities $p_{in}(u)$
• code bit sequence probabilities:

$$p_{out}(c) = \begin{cases}     p_{in}(\phi^{-1}(c)), & c \in C \\     0, & \text{else} \end{cases}$$

• conceptually simple, computationally infeasible

Bit-level

• given: info bit probabilities $p_{in}(u_k) \rightarrow p_{in}(u) = \prod_{k=1}^{K} p_{in}(u_k)$
• desired: code bit probabilities

$$p_{out}(c_n) = \sum_{\sim c_n} p_{out}(c) = \sum_{c \in C: c_n} p_{in}(\phi^{-1}(c))$$

• Main question: efficient implementation?
Soft Encoding: LLRs

System model:

![Diagram showing the system model with a binary source, noisy channel, and SISO encoder.]

Log-likelihood ratios (LLR)

- definition:

\[
L(u_k) = \log \frac{p_{\text{in}}(u_k = 0)}{p_{\text{in}}(u_k = 1)}, \quad L(c_n) = \log \frac{p_{\text{out}}(c_n = 0)}{p_{\text{out}}(c_n = 1)}
\]

- encoder input: \( L(u) = (L(u_1) \ldots L(u_K))^T \)

- encoder output: \( L(c) = (L(c_1) \ldots L(c_N))^T \)
HIHO versus SISO

HIHO is reversible: $u_1 \xrightarrow{\text{HIHO encoder}} c \xrightarrow{\text{HIHO decoder}} u_2 \equiv u_1$

SISO is irreversible: $L_1(u) \xrightarrow{\text{SISO encoder}} L(c) \xrightarrow{\text{SISO decoder}} L_2(u) \neq L_1(u)$

- except if $|L(u_k)| = \infty$ for all $k$
- however: $\text{sign}(L_1(u_k)) = \text{sign}(L_2(u_k))$

Post-sliced SISO identical to pre-sliced HIHO
Block Codes (1)

Binary \((N, K)\) block code \(C\) with generator matrix \(G \in \mathbb{F}_2^{N \times K}\)

**HIHO encoding:** \(c = Gu\), involves XOR/modulo 2 sum \(\oplus\)

**Statistics of XOR:**

\[
p(u_k \oplus u_l = 0) = p(u_k = 0)p(u_l = 0) + p(u_k = 1)p(u_l = 1)
\]

**Boxplus:** \(L(u_k \oplus u_l) \triangleq L(u_k) \boxplus L(u_l) = \frac{1 + e^{L(u_k)+L(u_l)}}{e^{L(u_k)} + e^{L(u_l)}}\)

- \(\boxplus\) is associative and commutative
- \(|L_1 \boxplus L_2| \leq \min\{|L_1|, |L_2|\}\)


**SISO encoder:** replace “\(\oplus\)” in HIHO encoder by “\(\boxplus\)”
Example: systematic (5, 3) block code

\[
\begin{pmatrix}
  c_1 \\
  c_2 \\
  c_3 \\
  c_4 \\
  c_5 \\
\end{pmatrix}
= \begin{pmatrix}
  1 & 0 & 0 \\
  0 & 1 & 0 \\
  0 & 0 & 1 \\
  1 & 1 & 0 \\
  0 & 1 & 1 \\
\end{pmatrix}
\begin{pmatrix}
  u_1 \\
  u_2 \\
  u_3 \\
\end{pmatrix}
= \begin{pmatrix}
  u_1 \\
  u_2 \\
  u_3 \\
  u_1 \oplus u_2 \\
  u_2 \oplus u_3 \\
\end{pmatrix}
\]
• Code \( C \) with given trellis: use **BCJR algorithm**

\[
\begin{align*}
\alpha_k(m) &= \sum_{m' = 0}^{M-1} \alpha_{k-1}(m') \gamma_k(m', m) \\
\beta_k(m) &= \sum_{m' = 0}^{M-1} \beta_{k+1}(m) \gamma_k(m, m')
\end{align*}
\]

\( p_{b}(c(j)_{k}) = \sum_{(m', m) \in A(j)} b \alpha_k(m') \gamma_k(m', m) \beta_k(m) \)
• Code $\mathcal{C}$ with given trellis: use **BCJR algorithm**

\[
\gamma_k(m', m) = P \{ S_{k+1} = m | S_k = m' \}
\]

\[
\alpha_k(m) = \sum_{m'=0}^{M-1} \alpha_{k-1}(m') \gamma_{k-1}(m', m)
\]

\[
\beta_k(m) = \sum_{m'=0}^{M-1} \beta_{k+1}(m') \gamma_k(m, m')
\]

\[
p_b(c_k^{(j)}) = \sum_{(m', m) \in A_b^{(j)}} \alpha_k(m') \gamma_k(m', m) \beta_{k+1}(m)
\]

---

Observe: only 2 transition probabilities per time instant

- Backward recursion $\beta_k(m)$ is rendered superfluous
• Observe: only 2 transition probabilities per time instant
  ▶ **Backward recursion** $\beta_k(m)$ is rendered superfluous

  ![Diagram showing backward recursion](image)

• **Simplified forward recursion encoder (FRE)**, reduces
  ▶ computational complexity
  ▶ memory requirements
  ▶ encoding delay

\[
s_{k+1}(m) = \sum_{m' \in B_m} s_k(m') \gamma_k(m', m)
\]

\[
p_b(c_k^{(j)}) = \sum_{(m', m) \in A^{(j)}_b} s_k(m') \gamma_k(m', m)
\]
Soft Encoding: Convolutional Codes (3)

- Special case: **non-recursive shift register encoder (SRE)**
  - soft encoder with shift register implementation
  - linear complexity with minimal memory requirements

- Example: \((7, 5)_8\) convolutional code
Soft Encoding: LDPC Codes (1)

- Sparse **parity check matrix** $H$ given: $v \in \mathcal{C}$ iff $H^T v = 0$

- Graphical representation of $H$: **Tanner graph**
  - bipartite graph with variable nodes and check nodes
  - let $\mathcal{V}$ denote the set of variable nodes

- Encoding: **iterative erasure filling**
  - matrix multiplication $Gu$ is infeasible for large block lengths
  - erasure pattern is known
Consider systematic code: \( c = (u^T p^T)^T \)

**Erasure channel:** \( L(c) = (L(u)^T L(p)^T)^T \mapsto (L(u)^T 0^T)^T \)

**SISO encoding = decoding ...**
- for the erasure channel (erasure filling) or, equivalently,
- w/o channel observation, but with prior soft information on \( u \)

Consider **special problem structure**
- yields efficient implementation
- (much) less complex than SISO decoding
- adjustable accuracy/complexity trade-off
• **Iterative erasure filling algorithm**
  1. find all check nodes involving a single erasure
  2. fill the erasures found in step 1
  3. repeat steps 1-2 until there are no more (recoverable) erasures
Soft Encoding: LDPC Codes (2)

- **Iterative erasure filling algorithm**
  1. find all check nodes involving a single erasure
  2. fill the erasures found in step 1
  3. repeat steps 1-2 until there are no more (recoverable) erasures

- **Encoding example:**

  ![Graphical representation](image)

  - information bit node
  - parity bit node
  - check node

- **Problem: stopping sets** ⇒ non-recoverable erasures
• **Definition:** \( S \subseteq V \) is a stopping set if all neighbors of \( S \) are connected to \( S \) at least twice

• Example:

![Diagram](image)
**Definition:** \( S \subseteq V \) is a stopping set if all neighbors of \( S \) are connected to \( S \) at least twice

**Example:**

**Solution:** modify \( \mathbf{H} \) such that stopping sets vanish

- constraints: code \( \mathcal{C} \) remains unchanged and \( \mathbf{H} \) remains sparse

1. Introduction
2. Soft channel encoding
3. Approximations
4. Applications
5. Conclusions and outlook
Approximations: Boxplus Operator

- Boxplus operator is used frequently in SISO encoding
  - Recall $a ⊙ b = \frac{1 + e^{a+b}}{e^a + e^b}$

- Approximation: $a ⊙ b \approx a \tilde{⊙} b = \text{sign}(a)\text{sign}(b) \min(|a|, |b|)$
  - small error if $|a| - |b|$ large
  - overestimates true result: $a \tilde{⊙} b \geq a ⊙ b$
  - suitable for hardware implementation
Approximations: Boxplus Operator

- **Boxplus operator** is used frequently in SISO encoding
  - Recall \( a \boxplus b = \frac{1 + e^{a+b}}{e^a + e^b} \)

- **Approximation:** \( a \boxplus b \approx a \tilde{\boxplus} b = \text{sign}(a)\text{sign}(b) \min(|a|, |b|) \)
  - small error if \(|a| - |b|\) large
  - overestimates true result: \( a \tilde{\boxplus} b \geq a \boxplus b \)
  - **suitable for hardware implementation**

- **Correction terms**
  - \( a \boxplus b = a \tilde{\boxplus} b + \log(1 + e^{-|a+b|}) - \log(1 + e^{-|a-b|}) \)
    - \(-\log(2) \leq\) additive correction \(\leq 0\)
  - \( a \boxplus b = a \tilde{\boxplus} b \cdot \left(1 - \frac{1}{\min(|a|, |b|)} \log \frac{1 + e^{-|a|-|b|}}{1 + e^{-|a|+|b|}}\right) \)
    - \(0 \leq\) multiplicative correction \(\leq 1\)
  - Store correction term in (small) **lookup table**

- **Decrease lookup table size by LLR clipping**
Approximations: Max-log Approximation

- **FRE**: perform **computation in log-domain** \((f^* = \log f)\)

\[
s_{k+1}^*(m) = \log \sum_{m' \in B_m} \exp(s_k^*(m') + \gamma_k^*(m', m))
\]

\[
p_b^*(y_k^{(j)}) = \log \sum_{(m', m) \in A_b^{(j)}} \exp(s_k^*(m') + \gamma_k^*(m', m))
\]

- **Approximation**: \(\log \sum_k \exp(a_k) \approx \max_k a_k \triangleq a_M\)
Approximations: Max-log Approximation

- **FRE**: perform **computation in log-domain** \((f^* = \log f)\)

\[
\begin{align*}
  s^*_{k+1}(m) &= \log \sum_{m' \in B_m} \exp(s^*_k(m') + \gamma^*_k(m', m)) \\
  p^*_b(y^{(j)}_k) &= \log \sum_{(m', m) \in A^{(j)}_b} \exp(s^*_k(m') + \gamma^*_k(m', m))
\end{align*}
\]

- **Approximation**: \(\log \sum_k \exp(a_k) \approx \max_k a_k \triangleq a_M\)

- **Correction term**: \(\log(e^a + e^b) = \max(a, b) + \log(1 + e^{-|a-b|})\)
  
    - nesting yields \(\log \sum_k \exp(a_k) = a_M + \log \sum_k \exp(a_k - a_M)\)

- **Correction term depends only on** \(|a - b| \Rightarrow \text{lookup table}\)
Outline

1. Introduction

2. Soft channel encoding

3. Approximations

4. Applications

5. Conclusions and outlook
Applications: Soft Re-encoding (1)

- **Soft network coding** (NC) for the two-way relay channel
  - users A and B exchange independent messages
  - relay R performs **network coding with soft re-encoding**
Applications: Soft Re-encoding (1)

- **Soft network coding** (NC) for the two-way relay channel
  - users A and B exchange independent messages
  - relay R performs **network coding with soft re-encoding**

- Transmission of quantized soft information is critical

Applications: Soft Re-encoding (2)

- BER simulation results
  - sym. channel conditions, R halfway between A, B, rate 1 bpcu, 256 info bits, 4 level quantization, 1 decoder iteration

- SNR gain of \( \sim 4.5 \text{ dB} \) at \( \text{BER} \approx 10^{-3} \) over hard NC
Applications: Convolutional Network Coding

- Physical layer NC for the multiple-access relay channel
Applications: Convolutional Network Coding

- Physical layer NC for the multiple-access relay channel

\[ x_A \]  
\[ x_B \]  
\[ x_R \]

- NC at relay

\[ \hat{c}_A \]  
\[ \hat{c}_B \]  
\[ \hat{c}_R \]  
\[ L(c_A) \]  
\[ L(c_B) \]  
\[ L_R \]  
\[ L(c_A) \]  
\[ L(c_B) \]  
\[ L_R \]  

hard NC  
soft NC  
convolutional NC
Outline

1. Introduction
2. Soft channel encoding
3. Approximations
4. Applications
5. Conclusions and outlook
Conclusions and Outlook

Conclusions:

• Efficient methods for soft encoding
• Approximations facilitate practical implementation
• Applications show usefulness of soft encoding

Outlook:

• Frequency domain soft encoding
• Code and transceiver design for physical layer NC
• Performance analysis of physical layer NC schemes
Conclusions and Outlook

**Conclusions:**
- Efficient methods for soft encoding
- Approximations facilitate practical implementation
- Applications show usefulness of soft encoding

**Outlook:**
- Frequency domain soft encoding
- Code and transceiver design for physical layer NC
- Performance analysis of physical layer NC schemes
Conclusions and Outlook

Conclusions:
- Efficient methods for soft encoding
- Approximations facilitate practical implementation
- Applications show usefulness of soft encoding

Outlook:
- Frequency domain soft encoding
- Code and transceiver design for physical layer NC
- Performance analysis of physical layer NC schemes

Thank you!