The Concept of Soft Channel Encoding and its Applications in Wireless Relay Networks

Gerald Matz

Institute of Telecommunications
Vienna University of Technology

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

Introduction: Basic Idea

Soft information processing

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

Soft-input soft-output (SISO) encoding

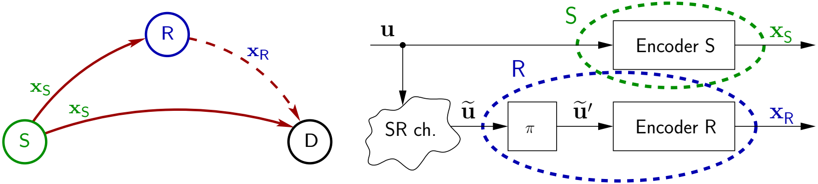
- extension of hard-input hard-output (HIHO) encoding
- **data only known in terms of probabilities**
- how to (efficiently) encode “noisy” data?

Applications

- physical layer network coding
- distributed (turbo) coding
- joint source-channel coding

Introduction: Motivation (1)

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

- H. Sneessens and L. Vandendorpe, "Soft decode and forward improves cooperative communications," in Proc. Workshop on Computational Advances in Multi-Sensor Adaptive Processing, 2005.
- P. Weitkemper, D. Wübben, V. Kühn, and K.D. Kammeyer, "Soft information relaying for wireless networks with errorprone source-relay link," in Proc. ITG Conference on Source and Channel Coding, 2008.

Transmission of soft information

- LLR quantization ⇒ **information bottleneck**

  N. Tishby, F. Pereira, and W. Bialek, "The information bottleneck method," in Proc. 37th Allerton Conference on Communication and Computation, 1999.
- quantizer labels & symbol mapping ⇒ **binary switching**

  K. Zeger and A. Gersho, "Pseudo-gray coding," IEEE Trans. Comm., vol. 38, no. 12, 1990.

Outline

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

Hard Encoding/Decoding Revisited

Notation

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

Linear binary channel code

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

Hard decoding:

- observed code bit sequence $\mathbf{c}' = \mathbf{c} \oplus \mathbf{e} = \phi(\mathbf{u}) \oplus \mathbf{e}$
- decoder: mapping ψ such that $\psi(\mathbf{c}')$ is "close" to $\mathbf{u}$

Soft Decoding Revisited

Word-level

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

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

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

Bit-level

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

Soft Encoding: Basics

Word-level

- given: info bit sequence probabilities $p_{\text{in}}(\mathbf{u})$
- code bit sequence probabilities:

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

- conceptually simple, computationally infeasible

Bit-level

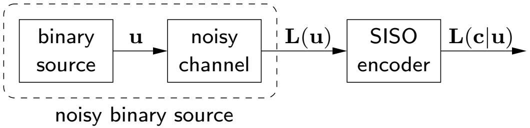
- given: info bit probabilities $p_{\text{in}}(u_k) \;\rightarrow\; p_{\text{in}}(\mathbf{u}) = \prod_{k=1}^{K} p_{\text{in}}(u_k)$
- desired: code bit probabilities

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

- **Main question:** efficient implementation?

Soft Encoding: LLRs

System model:



Log-likelihood ratios (LLR)

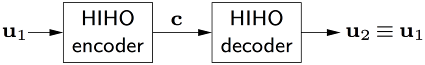
- definition:

$$L(u_k) = \log \frac{p_{\text{in}}(u_k = 0)}{p_{\text{in}}(u_k = 1)}, \qquad L(c_n) = \log \frac{p_{\text{out}}(c_n = 0)}{p_{\text{out}}(c_n = 1)}$$

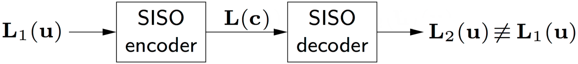
- encoder input: $\mathbf{L}(\mathbf{u}) = (L(u_1) \, \ldots \, L(u_K))^T$
- encoder output: $\mathbf{L}(\mathbf{c}) = (L(c_1) \, \ldots \, L(c_N))^T$

HIHO versus SISO

HIHO is reversible:

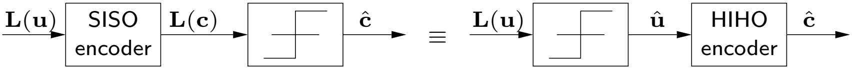


SISO is irreversible:



- 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-scliced HIHO



Block Codes (1)

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

HIHO encoding: $\mathbf{c} = \mathbf{G}\mathbf{u}$, 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: $L(u_k \oplus u_l) \triangleq L(u_k) \boxplus L(u_l) = \dfrac{1 + e^{L(u_k) + L(u_l)}}{e^{L(u_k)} + e^{L(u_l)}}$

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

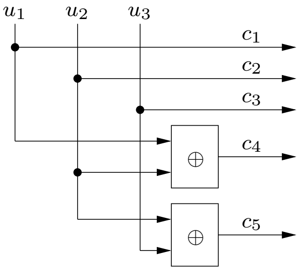
J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," IEEE Trans. Inf. Theory, vol. 42, no. 2, Feb. 1996.

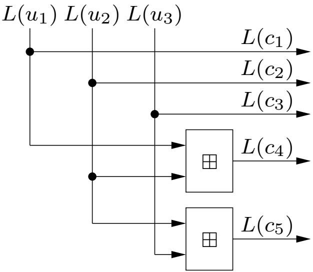
SISO encoder: replace "$\oplus$" in HIHO encoder by "$\boxplus$"

Block Codes (2)

Example: systematic (5, 3) block code

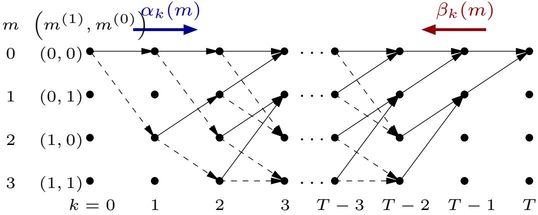
$$\underbrace{\begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{pmatrix}}_{\mathbf{c}} = \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}}_{\mathbf{G}} \underbrace{\begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix}}_{\mathbf{u}} = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_1 \oplus u_2 \\ u_2 \oplus u_3 \end{pmatrix}$$





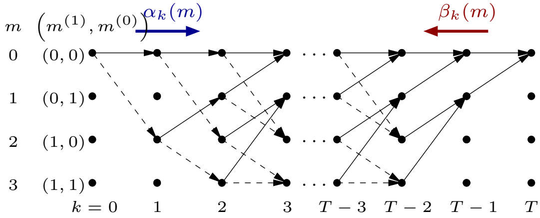
Soft Encoding: Convolutional Codes (1)

- Code $\mathcal{C}$ with given trellis: use **BCJR algorithm**



Soft Encoding: Convolutional Codes (1)

- Code $\mathcal{C}$ with given trellis: use **BCJR algorithm**



$$\gamma_k(m', m) = \mathsf{P}\left\{S_{k+1} = m | S_k = m'\right\}$$

$$\alpha_k(m) = \sum\nolimits_{m'=0}^{M-1} \alpha_{k-1}(m')\gamma_{k-1}(m', m)$$

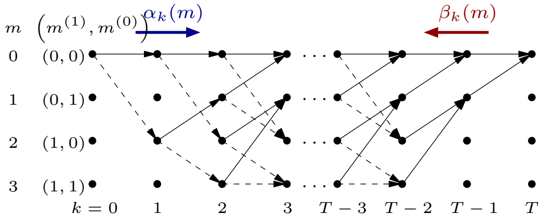
$$\beta_k(m) = \sum\nolimits_{m'=0}^{M-1} \beta_{k+1}(m')\gamma_k(m, m')$$

$$p_b\left(c_k^{(j)}\right) = \sum\nolimits_{(m',m)\in\mathcal{A}_b^{(j)}} \alpha_k(m')\gamma_k(m', m)\beta_{k+1}(m)$$

A. Winkelbauer and G. Matz, "On efficient soft-input soft-output encoding of convolutional codes," in Proc. ICASSP 2011

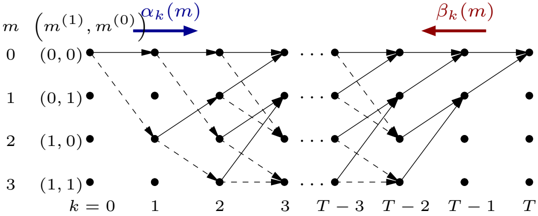
Soft Encoding: Convolutional Codes (2)

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



Soft Encoding: Convolutional Codes (2)

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



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

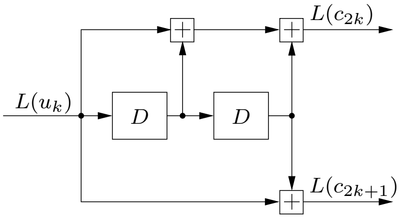
$$s_{k+1}(m) = \sum\nolimits_{m' \in \mathcal{B}_m} s_k(m')\gamma_k(m', m)$$

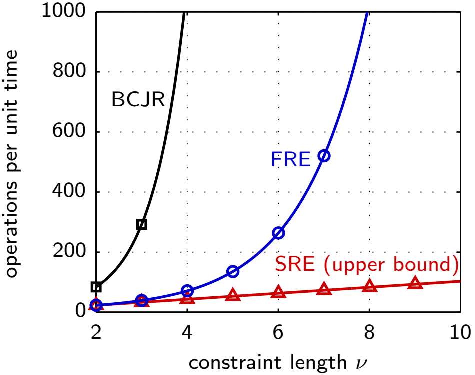
$$p_b\big(c_k^{(j)}\big) = \sum\nolimits_{(m',m) \in \mathcal{A}_b^{(j)}} s_k(m')\gamma_k(m', m)$$

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**

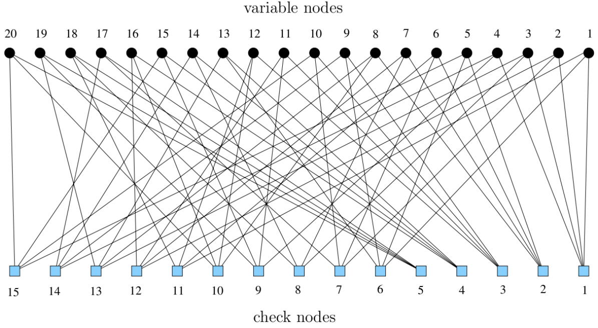
- Example: $(7,5)_8$ convolutional code





Soft Encoding: LDPC Codes (1)

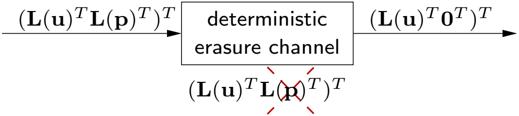
- Sparse **parity check matrix** $\mathbf{H}$ given: $\mathbf{v} \in \mathcal{C}$ iff $\mathbf{H}^T\mathbf{v} = \mathbf{0}$
- Graphical representation of $\mathbf{H}$: **Tanner graph**
  - bipartite graph with variable nodes and check nodes
  - let $\mathcal{V}$ denote the set of variable nodes



- Encoding: **iterative erasure filling**
  - matrix multiplication $\mathbf{Gu}$ is infeasible for large block lengths
  - **erasure pattern is known**

Soft Encoding: First Approach

Consider systematic code: $\mathbf{c} = (\mathbf{u}^T\ \mathbf{p}^T)^T$
Erasure channel: $\mathbf{L}(\mathbf{c}) = (\mathbf{L}(\mathbf{u})^T\ \mathbf{L}(\mathbf{p})^T)^T \mapsto (\mathbf{L}(\mathbf{u})^T\ \mathbf{0}^T)^T$



SISO encoding = decoding ...

- for the erasure channel (erasure filling) or, equivalently,
- w/o channel observation, but with prior soft information on $\mathbf{u}$

Consider **special problem structure**

- yields efficient implementation
- (much) less complex than SISO decoding
- adjustable accuracy/complexity trade-off

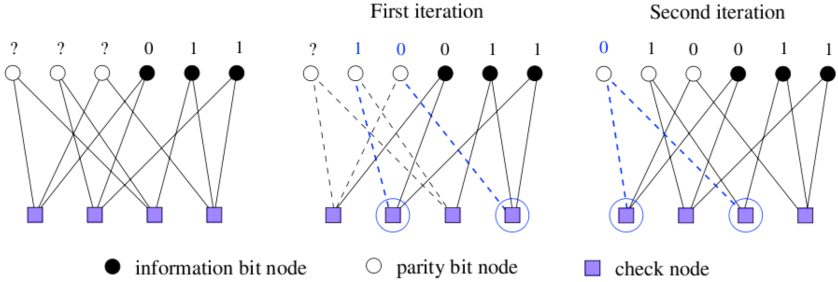
Soft Encoding: LDPC Codes (2)

- **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

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:

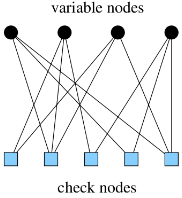


- **Problem: stopping sets** ⇒ non-recoverable erasures

Soft Encoding: LDPC Codes (3)

- **Definition:** $\mathcal{S} \subseteq \mathcal{V}$ is a stopping set if all neighbors of $\mathcal{S}$ are connected to $\mathcal{S}$ at least twice

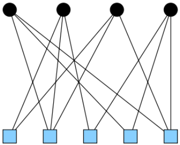
- Example:



Soft Encoding: LDPC Codes (3)

- **Definition:** $\mathcal{S} \subseteq \mathcal{V}$ is a stopping set if all neighbors of $\mathcal{S}$ are connected to $\mathcal{S}$ at least twice

- Example:



- **Solution: modify $\mathbf{H}$ such that stopping sets vanish**
  - constraints: code $\mathcal{C}$ remains unchanged and $\mathbf{H}$ remains sparse

M. Shaqfeh, N. Görtz, "Systematic Modification of Parity-Check Matrices for Efficient Encoding of LDPC Codes," in Proc. ICC 2007

Outline

Approximations: Boxplus Operator

- Boxplus operator is used frequently in SISO encoding
  - Recall $a \boxplus b = \dfrac{1 + e^{a+b}}{e^a + e^b}$
- **Approximation:** $a \boxplus b \approx a \mathbin{\widetilde{\boxplus}} b = \mathrm{sign}(a)\mathrm{sign}(b)\min(|a|, |b|)$
  - small error if $\big||a| - |b|\big|$ large
  - overestimates true result: $a \mathbin{\widetilde{\boxplus}} b \geq a \boxplus b$
  - **suitable for hardware implementation**

Approximations: Boxplus Operator

- Boxplus operator is used frequently in SISO encoding
  - Recall $a \boxplus b = \dfrac{1 + e^{a+b}}{e^a + e^b}$
- **Approximation:** $a \boxplus b \approx a \mathbin{\widetilde{\boxplus}} b = \mathrm{sign}(a)\mathrm{sign}(b)\min(|a|, |b|)$
  - small error if $\big||a| - |b|\big|$ large
  - overestimates true result: $a \mathbin{\widetilde{\boxplus}} b \geq a \boxplus b$
  - **suitable for hardware implementation**
- Correction terms
  - $a \boxplus b = a \mathbin{\widetilde{\boxplus}} b + \underbrace{\log(1 + e^{-|a+b|}) - \log(1 + e^{-|a-b|})}_{-\log(2) \,\leq\, \text{additive correction} \,\leq\, 0}$
  - $a \boxplus b = a \mathbin{\widetilde{\boxplus}} b \cdot \underbrace{\left(1 - \dfrac{1}{\min(|a|, |b|)} \log \dfrac{1 + e^{-\big||a|-|b|\big|}}{1 + e^{-\big||a|+|b|\big|}}\right)}_{0 \,\leq\, \text{multiplicative correction} \,\leq\, 1}$
  - Store correction term in (small) **lookup table**
- **Decrease lookup table size by LLR clipping**

Approximations: Max-log Approximation

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

$$s^*_{k+1}(m) = \log \sum\nolimits_{m' \in \mathcal{B}_m} \exp\big(s^*_k(m') + \gamma^*_k(m', m)\big)$$
$$p^*_b\big(y_k^{(j)}\big) = \log \sum\nolimits_{(m',m) \in \mathcal{A}_b^{(j)}} \exp\big(s^*_k(m') + \gamma^*_k(m', m)\big)$$

- **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** $(f^* = \log f)$

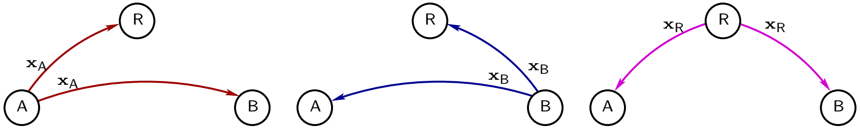
$$s^*_{k+1}(m) = \log \sum\nolimits_{m' \in \mathcal{B}_m} \exp\big(s^*_k(m') + \gamma^*_k(m', m)\big)$$
$$p^*_b\big(y^{(j)}_k\big) = \log \sum\nolimits_{(m',m) \in \mathcal{A}^{(j)}_b} \exp\big(s^*_k(m') + \gamma^*_k(m', m)\big)$$

- **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 $|a - b| \Rightarrow$ **lookup table**

Outline

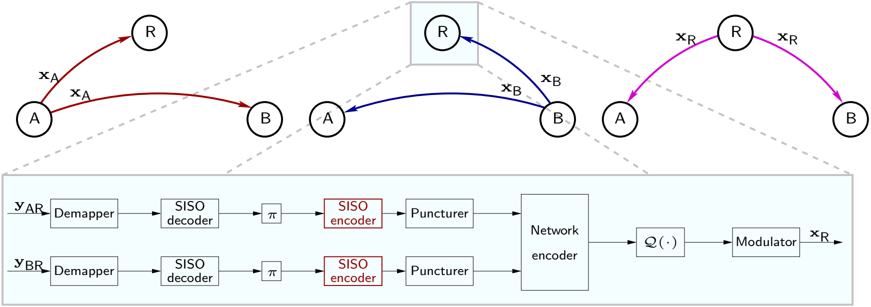
Applications: Soft Re-encoding (1)

- **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**

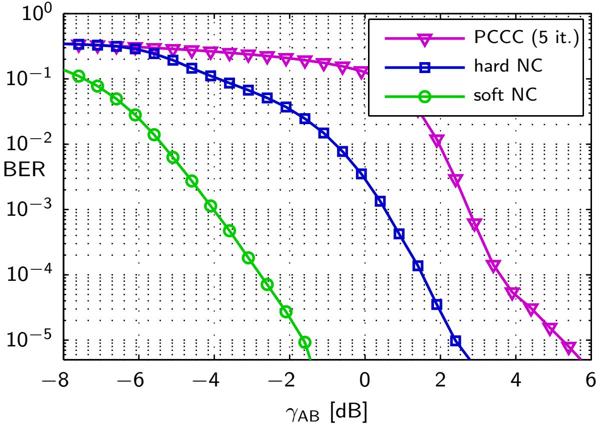


- **Transmission of quantized soft information is critical**

A. Winkelbauer and G. Matz, "Soft-Information-Based Joint Network-Channel Coding for the Two-Way Relay Channel," submitted to NETCOD 2011

Applications: Soft Re-encoding (2)

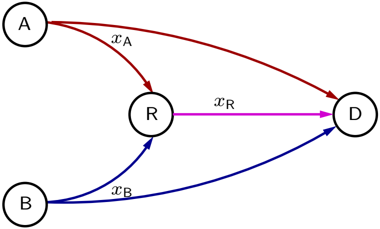
- BER **simulation results**
  - sym. channel conditions, R halfway between A, B, rate 1 bpcu, 256 info bits, 4 level quantization, 1 decoder iteration



- **SNR gain of ~ 4.5 dB at $\mathrm{BER} \approx 10^{-3}$ over hard NC**

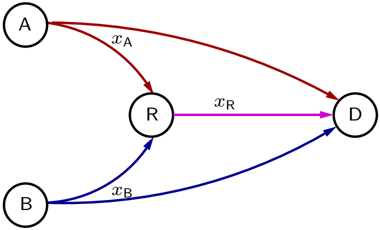
Applications: Convolutional Network Coding

- Physical layer NC for the multiple-access relay channel

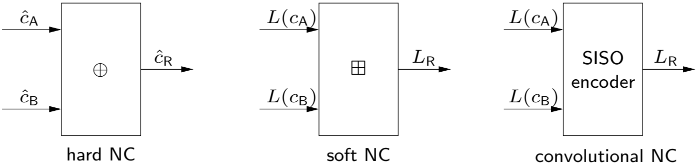


Applications: Convolutional Network Coding

- Physical layer NC for the multiple-access relay channel



- NC at relay



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

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!