Partitioned Successive-Cancellation List Decoding of Polar Codes

Seyyed Ali Hashemi[⋆], Alexios Balatsoukas-Stimming[†], Pascal Giard[⋆], Claude Thibeault[⋄], Warren J. Gross[⋆]

[⋆]McGill University, Montréal, Québec, Canada
[†]École polytechnique fédérale de Lausanne, Lausanne, Switzerland
[⋄]École de technologie supérieure, Montréal, Québec, Canada

March 23, 2016

What is the problem?

- Polar codes are state-of-the-art codes with interesting properties
- Successive-Cancellation List (SCL) decoding can outperform LDPC codes
- SCL requires large amount of memory
  - High memory requirement translates into high area occupation

In this talk:

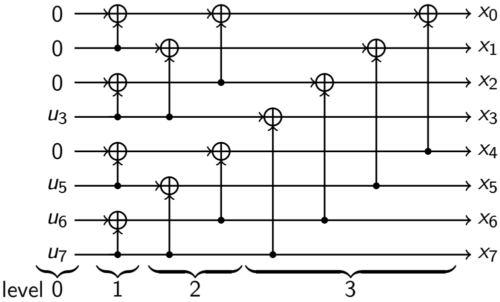
We reduce the memory usage of SCL
and
improve its performance!

Polar Codes

- First family of codes which can provably achieve the channel capacity with explicit construction and low-complexity decoding[1]
- The encoding process consists of recursive application of a linear transformation of $G = \left[\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right]$ to get the generator matrix $G^{\otimes n}$
  - The channels are then sorted as being either *good* or *bad*
  - As the number of channels increases, the fraction of good channels tends to the channel capacity
- Different decoding schemes are available:
  - Successive-Cancellation (SC)
  - SC List

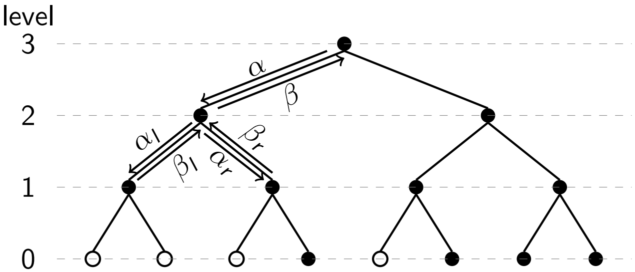
[1] E. Arıkan, "Channel polarization: A method for constructing capacity achieving codes for symmetric binary input memoryless channels," *Information Theory, IEEE Transactions on*, vol. 55, no. 7, pp. 3051-3073, July 2009.

Polar Encoding



- For a rate $\frac{K}{N}$ code $\mathcal{P}(N, K)$, the K best channels are found and the information bits are assigned to those channels
- The $N - K$ worse channels are set to a predefined value (usually 0). These are called *frozen bits* $\mathcal{F}$

Successive-Cancellation Decoding



$$\alpha_l[i] = \text{sgn}(\alpha[i])\text{sgn}(\alpha[i+2^{s-1}])\min(|\alpha[i]|, |\alpha[i+2^{s-1}]|),$$

$$\alpha_r[i] = \alpha[i+2^{s-1}] + (1-2\beta_l[i])\alpha[i],$$

$$\beta[i] = \begin{cases} \beta_l[i] \oplus \beta_r[i], & \text{if} \quad i < 2^{s-1} \\ \beta_r[i+2^{s-1}], & \text{otherwise,} \end{cases}$$

SC List Decoding

- SC decoding

$$\hat{u}_i = \begin{cases} 0, & \text{if } i \in \mathcal{F} \text{ or } \alpha_i \geq 0, \\ 1, & \text{otherwise.} \end{cases}$$

- SCL decoding

$$\hat{u}_i^l = \begin{cases} 0, & \text{if } i \in \mathcal{F} \\ 0 \text{ or } 1, & \text{otherwise.} \end{cases}$$

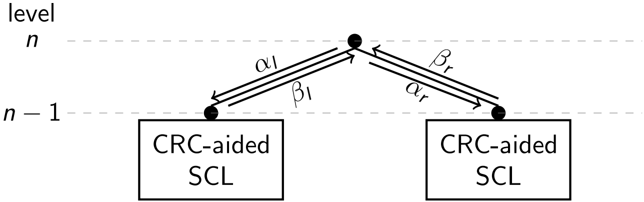
$$\mathrm{PM}_i^l = \begin{cases} \mathrm{PM}_{i-1}^l, & \text{if } \hat{u}_i^l = \frac{1}{2}\left(1 - \operatorname{sgn}\left(\alpha_i^l\right)\right), \\ \mathrm{PM}_{i-1}^l + |\alpha_i^l|, & \text{otherwise,} \end{cases}$$

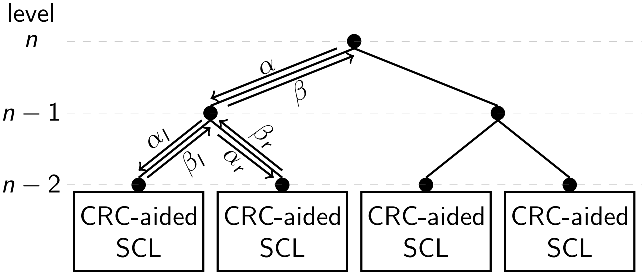
  - Cyclic Redundancy Check (CRC) code can be used to help SCL find the correct candidate

Partitioned SCL Decoding

- The decoder tree is broken into partitions (subtrees)
- SCL decoding is performed only on the partitions
- Standard SC rules are applied to the remainder of the decoding tree
- Each partition outputs a single candidate codeword which is selected with the help of a CRC and then sent to the next partition
- The decoding process starts with the standard SC update rules
- The decoder does not require memory to store L entire trees of internal LLRs
- Only L copies of the partitions on which SCL decoding is performed are stored

Partitioned SCL Decoding





Memory Requirements

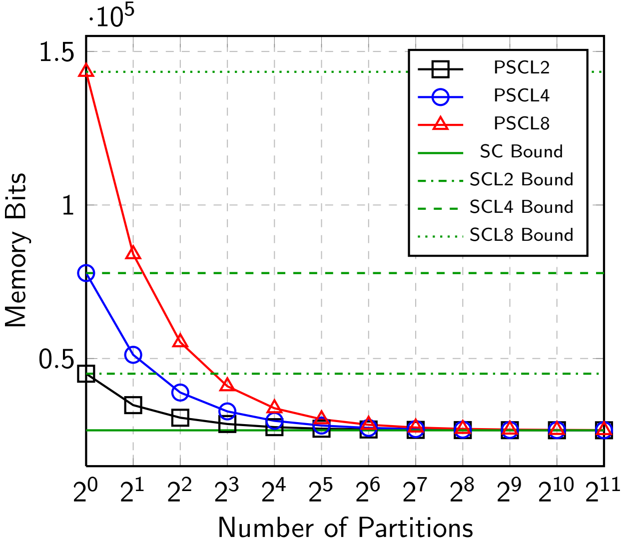
$$M_{\text{SC}} = \underbrace{(2N-1)\,Q_\alpha}_{\alpha\ \text{(LLR values)}} + \underbrace{N-1}_{\beta\ \text{(partial sums)}},$$

$$M_{\text{SCL}} = \underbrace{(N+(N-1)\,L)\,Q_\alpha}_{\alpha\ \text{(LLR values)}} + \underbrace{LQ_{\text{PM}}}_{\text{path metrics}} + \underbrace{(2N-1)\,L}_{\beta\ \text{(partial sums)}},$$

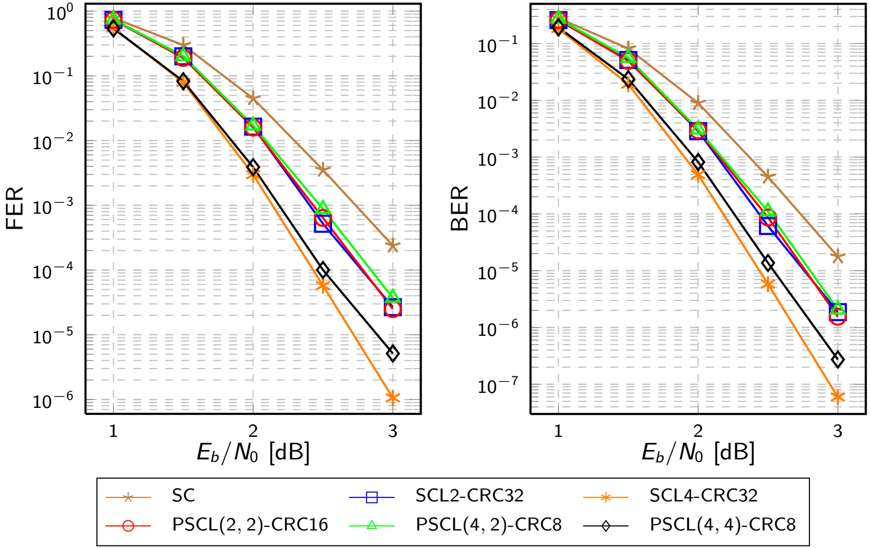
$$M_{\text{PSCL}} = \underbrace{\left(\sum_{k=0}^{P-1}\frac{N}{2^k} + \left(\frac{N}{2^{P-1}}-1\right)L\right)Q_\alpha}_{\alpha\ \text{(LLR values)}} + \underbrace{LQ_{\text{PM}}}_{\text{path metrics}}$$

$$+ \underbrace{\sum_{k=1}^{P-2}\frac{N}{2^k} + \left(\frac{N}{2^{P-2}}-1\right)L}_{\beta\ \text{(partial sums)}}$$

Memory Requirements



Performance Results $\mathcal{P}(2048, 1024)$



Implementation Results $\mathcal{P}(2048, 1024)$

Algorithm	Total (mm^2)	Memory (mm^2)
SC	0.723	0.413
SCL2-CRC32	1.563	0.702
SCL4-CRC32	3.075	1.214
PSCL$(2, 2)$-CRC16	1.189	0.540
PSCL$(4, 2)$-CRC8	0.909	0.415
PSCL$(4, 4)$-CRC8	1.356	0.543

Conclusions

- We proposed a novel PSCL decoding algorithm for polar codes
- The code is broken into partitions and each partition is decoded with a CRC-aided SCL decoder
- The memory requirements of PSCL decoder is significantly smaller than that of an SCL decoder
- At equivalent error-correction performance, PSCL leads to memory and total area savings of 41% and 42% compared to a similar SCL decoder implementation
- PSCL enables a coding gain of approximately 0.25 dB at a BER of 10^{-5} while occupying 13% less total area than the SCL decoder

In short:

PSCL achieves better performance *and* reduces memory usage at the same time!

Thank you!