Stepped List Decoding of Polar Codes

Mohammad Rowshan
Emanuele Viterbo

Dec 3-7, 2018
Int'l Symposium on Turbo Codes
& Iterative Info. Processing

Outline

Polar Codes [Arıkan, 2008]

Polar Codes:

- are provably capacity-achieving codes,
- have explicit construction,
- have replaced CC in 5G eMBB control channels.

For a polar code $PC(N, K, \mathcal{A})$:

- $N(=2^n)$: block-length,
- K: number of information bits,
- A: index set of the information/free bits

The generator matrix is $G_N = B_N G_2^{\otimes n}$, where:

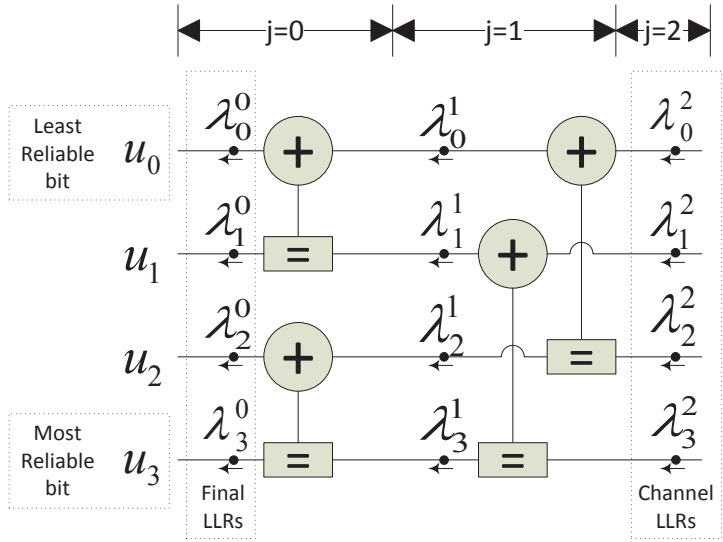
- $G_2 \triangleq \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$,
- B_N is an $N \times N$ bit-reversal permutation matrix, and
- $(\cdot)^{\otimes n}$ denotes the n-th Kronecker power.

Successive Cancellation (SC) Decoding

- A sub-optimal greedy tree search
- The information bits are estimated by hard decision:

$$\hat{u}_i = \begin{cases} 0 & \lambda_i^0 = \ln \frac{P(Y, \hat{u}_0^{i-1} | \hat{u}_i = 0)}{P(Y, \hat{u}_0^{i-1} | \hat{u}_i = 1)} > 0, \\ 1 & \text{otherwise} \end{cases}$$

- if $i \notin A$, regardless of λ_i^0, $\hat{u}_i$ is set as a frozen bit, i.e. $\hat{u}_i = 0$

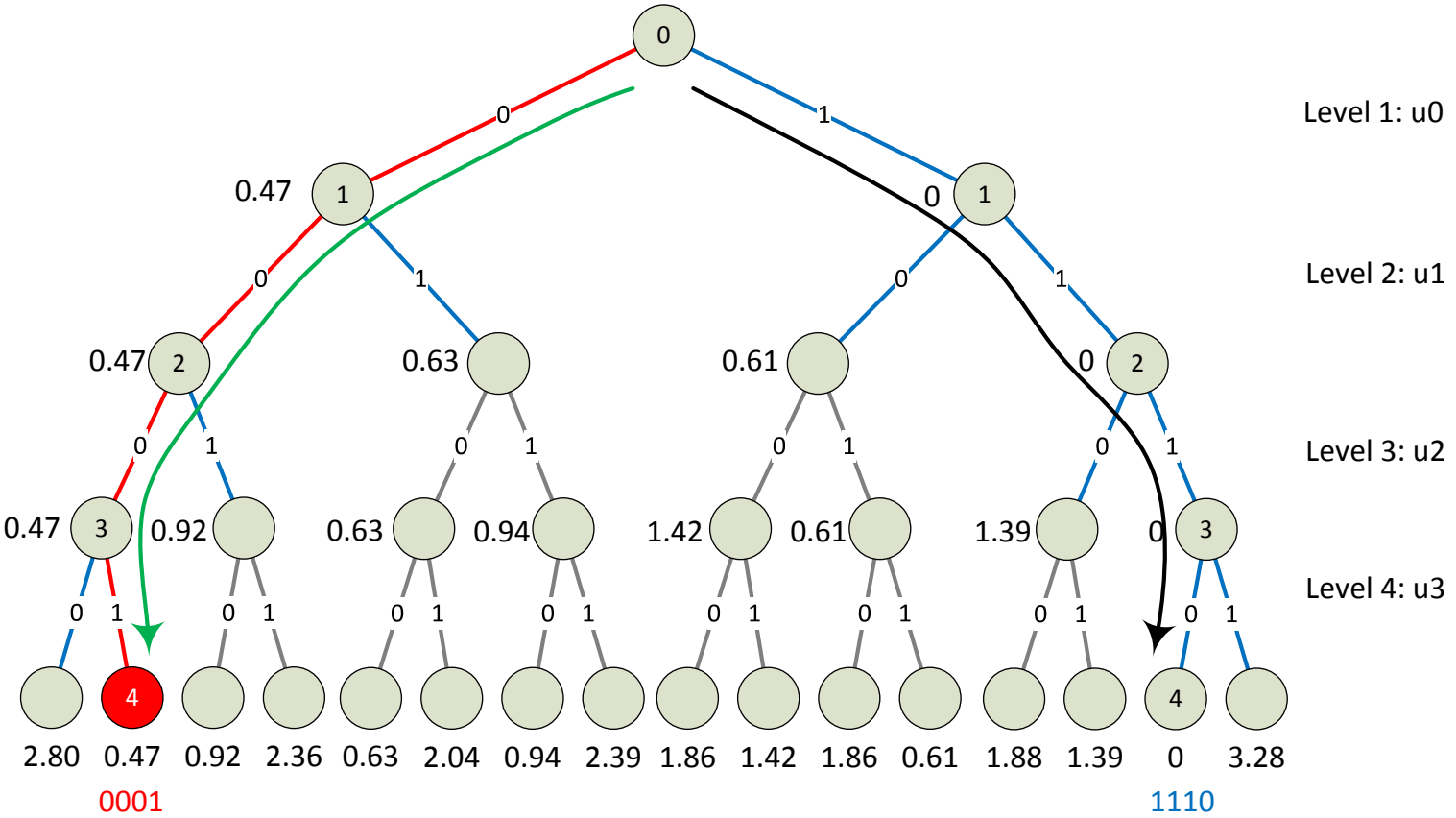


The internal LLRs are approximated by the following f and g functions:

$$f(\lambda_a, \lambda_b) = \text{sgn}(\lambda_a) \cdot \text{sgn}(\lambda_b) \cdot \min(|\lambda_a|, |\lambda_b|)$$

$$g(\lambda_a, \lambda_b, \hat{\beta}) = (-1)^{\hat{\beta}} \lambda_a + \lambda_b$$

SC Decoding vs SCL Decoding



List Size (L) = 2, —— SC/SCL solution, —— CRC-aided SCL solution

SC List Decoding [Tal & Vardy, 2011]

- Estimating every bit by considering both possible values: 0, 1
- L most reliable paths remains at each decision step.
- LLR-based path metric [Balatsoukas et al., 2014] used to measure the reliability of the paths:

$$PM_l^{(i)} = \begin{cases} PM_l^{(i-1)} + |\lambda_i^0[l]| & \text{if } \hat{u}_i[l] \neq \frac{1}{2}(1-\text{sgn}(\lambda_i^0[l])) \\ PM_l^{(i-1)} & \text{otherwise} \end{cases}$$

- When the SCL decoder fails, the correct path might still be in the list: Adding r CRC bits to the information bits as outer coding can assist in finding the correct path among the L final paths

Question: Do we really have to use a fixed list size?

List decoding provides a good error correction performance while requires large memory and has high computational complexity.

Solutions:

- List/Tree Pruning [Chen, 2014]
- Multi-CRC/Partitioned Decoding [Guo, 2016], [Hashemi, 2016]

Alternative Solution:

Finding a way to answer the above question by **NO**.

Evolution of path metrics (PMs) within the list throughout list decoding provides a good understanding toward the alternative solution.

Path Metric Range (PMR)

Definition 1: *Path metric range* is defined as

$$PMR_i = PM_L^{(i)} - PM_1^{(i)}$$

where $i \in \{0, 1, .., N-1\}$, assuming that the path metrics are sorted in ascending order $PM_1^{(i)} < PM_2^{(i)} < ... < PM_L^{(i)}$.

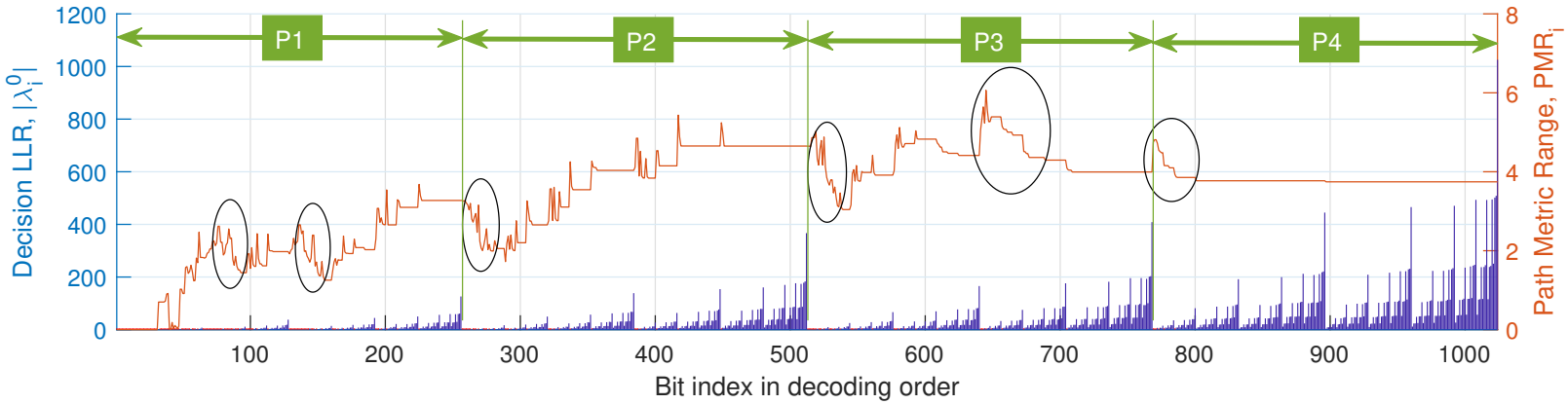
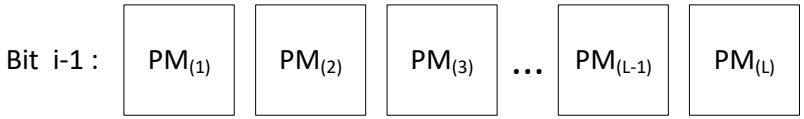


Figure 1: LLR values and PMR curves for N=1024 & R=0.8, L=32

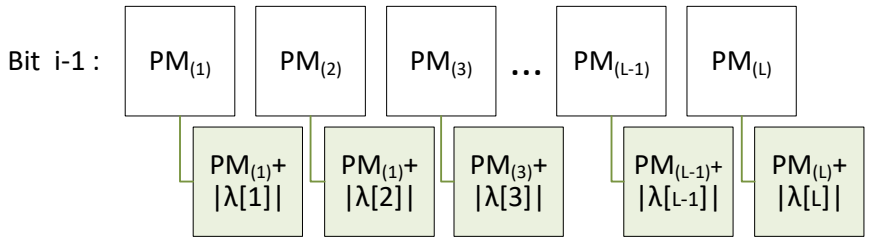
Why PMR drops?

Lemma: If u_i is an unfrozen bit ($i \in \mathcal{A}$), and $|\lambda_0^i[l]| < PMR_{i-1}$ for all l, then $PMR_i < PMR_{i-1}$.

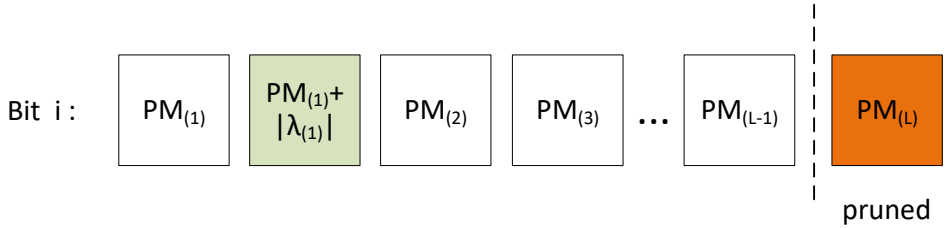
Proof: Assuming $PM_{(1)}^{(i-1)} < \ldots < PM_{(L)}^{(i-1)}$.



After splitting the paths at step/bit i, the $2L$ path metrics are:



Since $|\lambda_0^i[l]| < PMR_{i-1}$, then $PM_1^{(i-1)} + |\lambda_0^i[1]| < PM_L^{(i-1)}$, and



As a result, $PM_L^{(i)} < PM_L^{(i-1)}$, then $PMR_i < PMR_{i-1}$.

Crucial Bits in the Partitions

Definition 2: *Crucial bits* [Zhang, 2017] are defined as the unfrozen bits with $|\lambda_0^i[l]| < PMR_{i-1}$.
Crucial bits are distributed in the partitions. This distribution plays an important role.
Definition 3: *Partitions* are defined as the sub-trees of the decoding tree [Hashemi, 2016], associated with code's sub-blocks of length 2^m, $m < n$.The j-th partition and its associated sub-block are denoted by P_j, for $0 \leq j \leq 2^{n-m} - 1$.
S^j denotes the set of indices of crucial bits in the j-th partition.

Movement of Correct Path

Some sampled movements of correct path in the list, representing different scenarios, for PC(1024,820) and $L = 16$

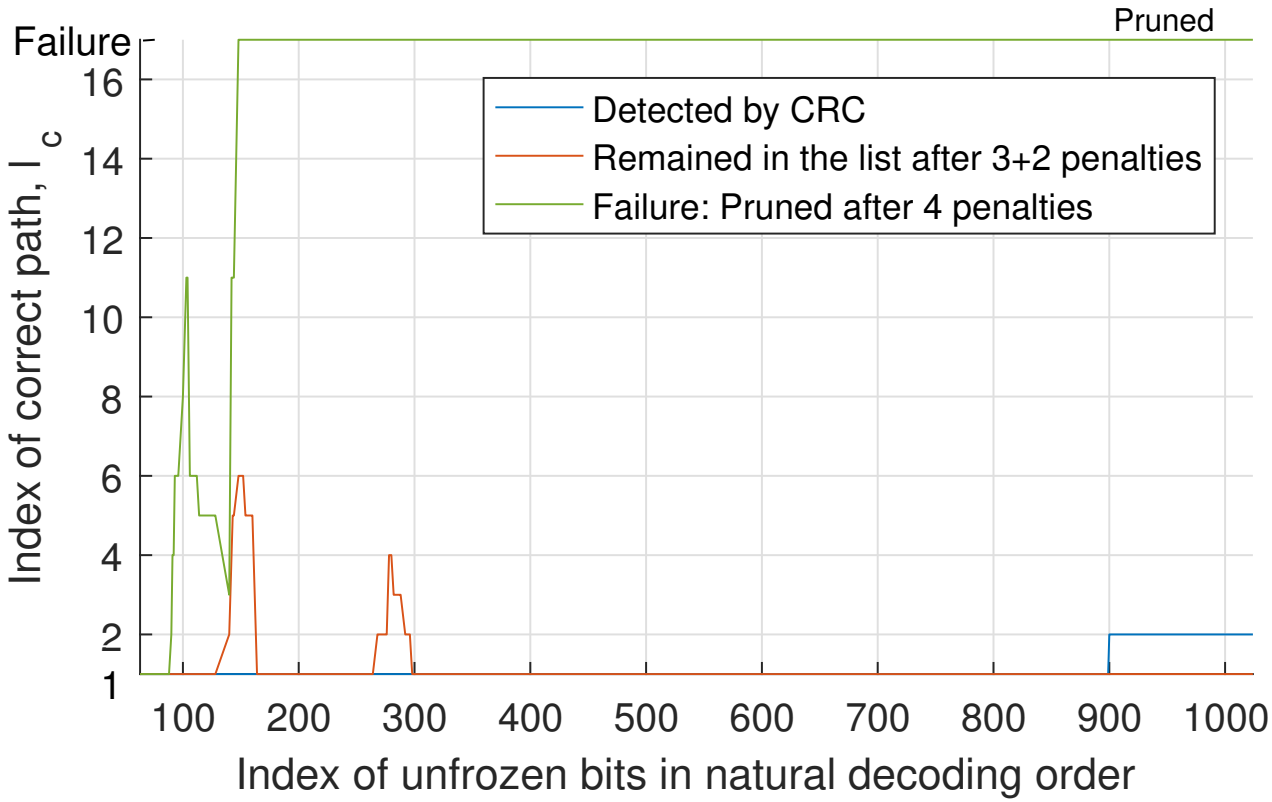


Figure 2: Movement of correct path throughout decoding

Application of PMR: Stepped List Decoding

- $PMR \propto L$
- if $u_i \in S^{p1}, u_j \in S^{p2}$ s.t. $PMR_i \ll PMR_j$, then we can find $L' < L$ such that $|\lambda_0^j[l]| < PMR'_j < PMR_j$ for all l,
- **Conclusion:** Reducing the list size L in the partition with high mean *PMR* does not affect the *error correction performance*, if the new list size L' is chosen optimally.
- **Method:** Local list size L'_j can be determined by:

$$\log_2(L'_j) \propto \frac{1}{PMR_{avg}^{(j)}} = \frac{|S^j|}{\sum_{S^j} PMR_i}$$

- **Examples** (for 4 partitions):

	$(PMR_{avg}^{(j)})_{1\le j\le 4}$	(L_1, L_2, L_3, L_4)	When $L = 2^t = 32$
PC(1024,820)	(2, 3, 4, 4)	$(2^t, 2^{t-1}, 2^{t-2}, 2^{t-2})$	(32, 16, 8, 8)
PC(1024,512)	(8, 6, 6, 9)	$(2^{t-1}, 2^t, 2^t, 2^{t-2})$	(16, 32, 32, 8)
PC(1024,256)	(-, 38, 23, 24)	$(2, 2^{t-1}, 2^t, 2^t)$	(2, 16, 32, 32)

In this scheme, the list size changes in a stepwise manner.

Example: PC(1024,820) - Path Memory

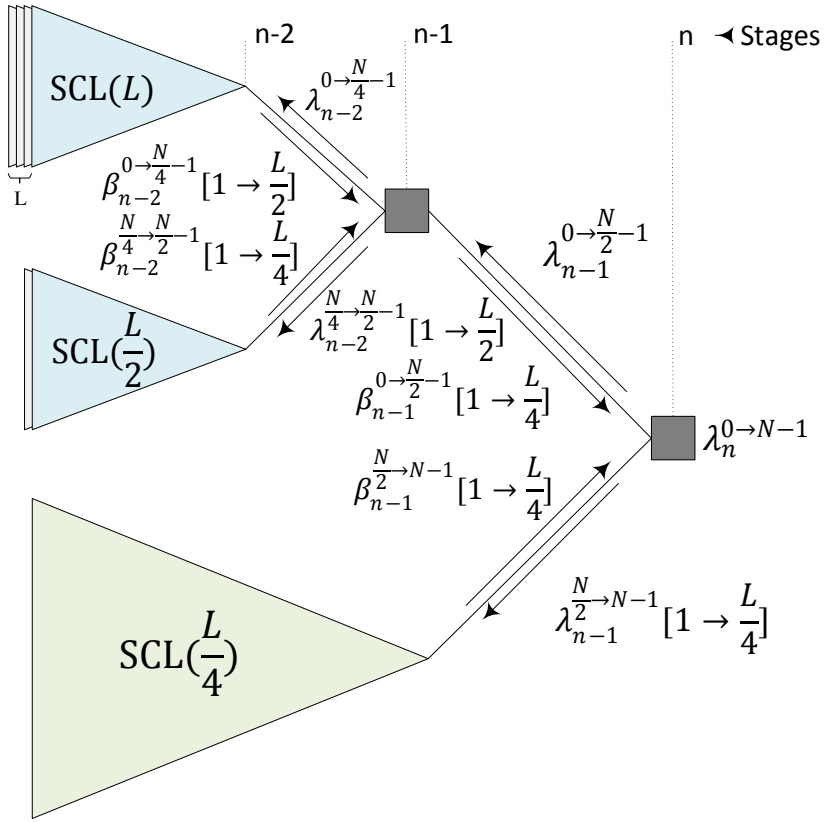


Figure 3: Decoding Tree

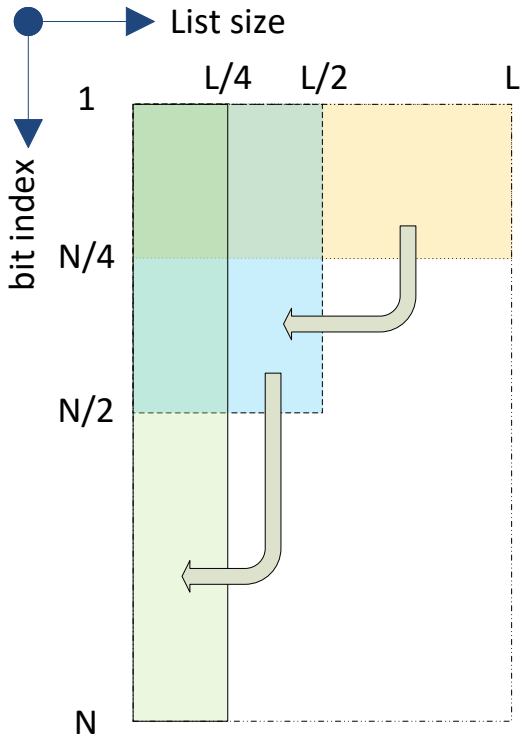


Figure 4: Path Memory

Example: PC(1024,820) - Complexity and Memory Reductions

Since computational complexity and memory requirement are proportional to list size L:

- Complexity $(L \cdot N \cdot \log N)$ reduces as N is divided into 4, 8 partitions with different list sizes, L_i.

$$\underbrace{L\frac{N}{4}\log_2 N}_{\text{1st partition}} + \underbrace{\frac{L}{2}\frac{N}{4}\log_2 N}_{\text{2nd partition}} + \underbrace{\frac{L}{4}\frac{N}{2}\log_2 N}_{\text{3rd/4th partition}} = \frac{1}{2}LN\log_2 N \tag{1}$$

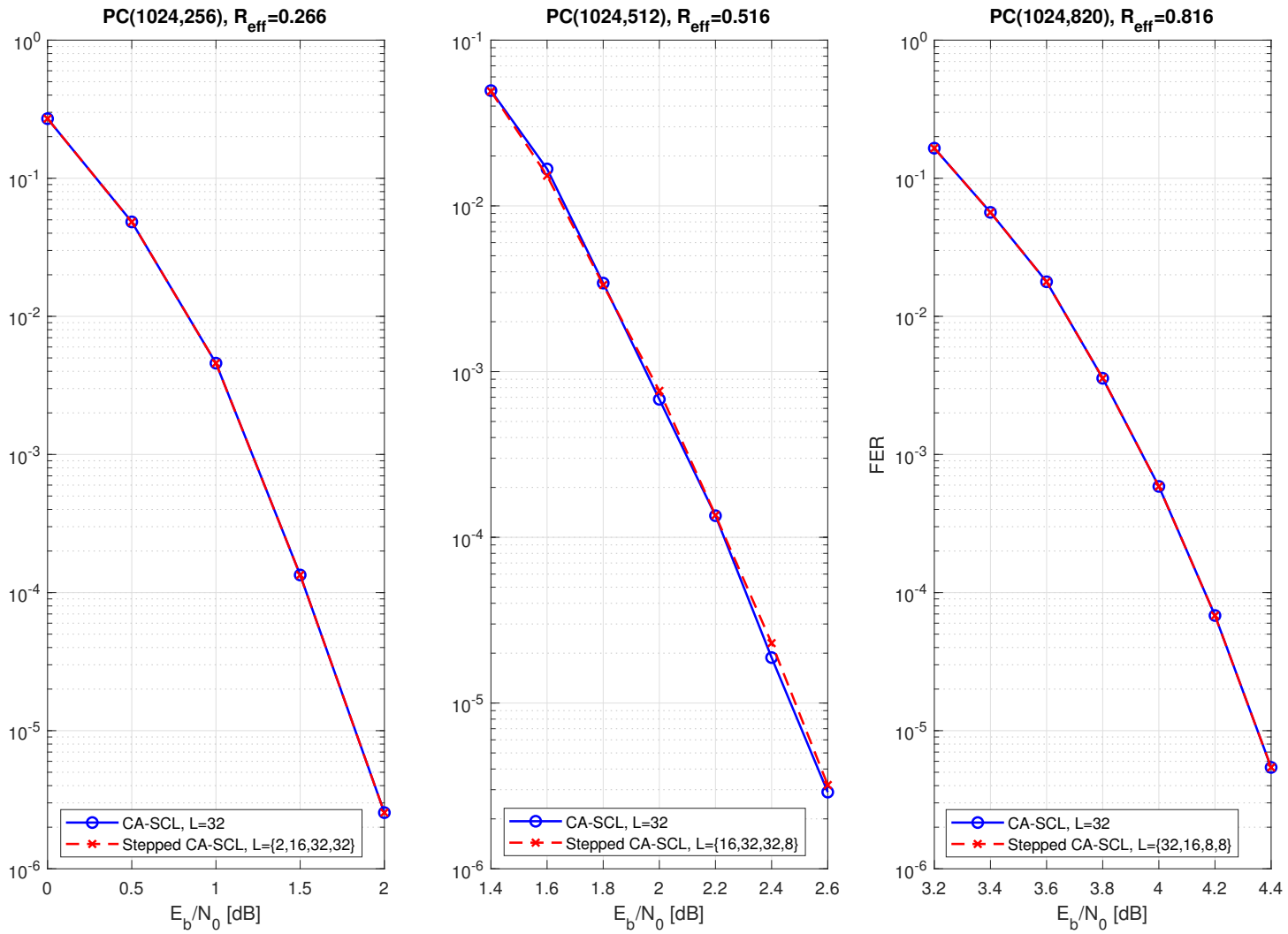
- LLR memory and path memory reduce in the same way.

$$M_{SCL} = \underbrace{L(N-1)Q_i}_{\text{Internal LLRs}} + \underbrace{L(N-1)}_{\text{Partial sums}} \tag{2}$$

$$M_{SCL-Stepped} = \underbrace{\left(L(\frac{N}{4}-1) + \frac{L}{2}(\frac{N}{4}) + \frac{L}{4}(\frac{N}{2})\right)Q_i}_{\text{Internal LLRs}}$$
$$+ \underbrace{\left(L(\frac{N}{4}-1) + \frac{L}{2}(\frac{N}{4}) + \frac{L}{4}(\frac{N}{2})\right)}_{\text{Partial sums}} = \underbrace{L(\frac{1}{2}N-1)Q_i}_{\text{Internal LLRs}} + \underbrace{L(\frac{1}{2}N-1)}_{\text{Partial sums}} \tag{3}$$

where Q_i is the number of bits used in quantization of internal LLRs

Performance of Stepped CA-SCLD vs Conv. CA-SCLD



Summary

- Evolution of PMs in the list throughout decoding provided an instrumental understanding.
- Stepped list decoding is built on the concept of path metric range (PMR),
- Advantages of stepped scheme:
  - Simplicity of the scheme,
  - No computational overhead,
  - No change in the effective code rate,
  - Significant reduction in computational complexity and memory requirement,
  - Reductions are SNR-independent,
  - Almost no performance degradation.