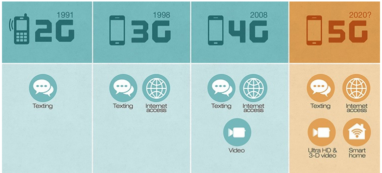
Partitioned Successive-Cancellation Flip Decoding of Polar Codes

Furkan Ercan, Carlo Condo, Seyyed Ali Hashemi, Warren J. Gross
Department of Electrical and Computer Engineering
McGill University
Montréal, Québec, Canada

May 22, 2018

Motivation



- **Polar Codes** provably achieve channel capacity
  - Adopted in **5G eMBB** control channel
  - Being considered for **5G URLLC & mMTC** channels
- **5G** standardization targets
  - Improved error-correction performance, T/P
  - Low power/energy consumption

[1] E. Arıkan. ”Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels,” IEEE Transactions on Information Theory, vol. 55, no. 7, pp. 30513073, July 2009.

Motivation

- Successive Cancellation **(SC)** decoding

Motivation

- Successive Cancellation **(SC)** decoding
  - Mediocre error-correction performance

Motivation

- Successive Cancellation **(SC)** decoding
  - Mediocre error-correction performance
- Successive Cancellation List **(SC-List)** decoding

Motivation

- Successive Cancellation **(SC)** decoding
  - Mediocre error-correction performance
- Successive Cancellation List **(SC-List)** decoding
  - ✓ Improved error-correction performance

Motivation

- Successive Cancellation **(SC)** decoding
  - Mediocre error-correction performance
- Successive Cancellation List **(SC-List)** decoding
  - ✓ Improved error-correction performance
  - ✗ Increased latency, area, power consumption

Motivation

- Successive Cancellation **(SC)** decoding
  - Mediocre error-correction performance
- Successive Cancellation List **(SC-List)** decoding
  - ✓ Improved error-correction performance
  - ✗ Increased latency, area, power consumption
- Successive Cancellation Flip **(SC-Flip)** decoding

Motivation

- Successive Cancellation **(SC)** decoding
  - Mediocre error-correction performance
- Successive Cancellation List **(SC-List)** decoding
  - ✓ Improved error-correction performance
  - ✗ Increased latency, area, power consumption
- Successive Cancellation Flip **(SC-Flip)** decoding
  - ✓ Error-correction performance comparable to SC-List

Motivation

- Successive Cancellation **(SC)** decoding
  - Mediocre error-correction performance
- Successive Cancellation List **(SC-List)** decoding
  - ✓ Improved error-correction performance
  - ✗ Increased latency, area, power consumption
- Successive Cancellation Flip **(SC-Flip)** decoding
  - ✓ Error-correction performance comparable to SC-List
  - ✓ Implementation complexity close to that of SC

Motivation

- Successive Cancellation **(SC)** decoding
  - Mediocre error-correction performance
- Successive Cancellation List **(SC-List)** decoding
  - ✓ Improved error-correction performance
  - ✗ Increased latency, area, power consumption
- Successive Cancellation Flip **(SC-Flip)** decoding
  - ✓ Error-correction performance comparable to SC-List
  - ✓ Implementation complexity close to that of SC
  - ✓ Average latency converges to that of SC

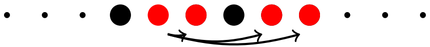
SC-Flip Decoding: Idea

- Classifying erroneous decisions into:
  - Channel-induced errors
  - Propagated errors due to a previous error

[2] O. Afisiadis, A. Balatsoukas-Stimming and A. Burg, ”A low-complexity improved successive cancellation decoder for polar codes,” 2014 48th Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, 2014, pp. 2116-2120.

SC-Flip Decoding: Idea

- Classifying erroneous decisions into:
  - Channel-induced errors
  - Propagated errors due to a previous error
- First error (E_1) is always due to channel
- Improved error-correction if first error is avoided!



[2] O. Afisiadis, A. Balatsoukas-Stimming and A. Burg, ”A low-complexity improved successive cancellation decoder for polar codes,” 2014 48th Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, 2014, pp. 2116-2120.

SC-Flip Decoding: Idea

- Classifying erroneous decisions into:
  - Channel-induced errors
  - Propagated errors due to a previous error
- First error (E_1) is always due to channel
- Improved error-correction if first error is avoided!
- Goal: Locate and correct the first erroneous decision

[2] O. Afisiadis, A. Balatsoukas-Stimming and A. Burg, ”A low-complexity improved successive cancellation decoder for polar codes,” 2014 48th Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, 2014, pp. 2116-2120.

SC-Flip Decoding: Idea

- Classifying erroneous decisions into:
  - Channel-induced errors
  - Propagated errors due to a previous error
- First error (E_1) is always due to channel
- Improved error-correction if first error is avoided!
- Goal: Locate and correct the first erroneous decision
  - SC decoding is supported by a CRC
  - Multiple SC iterations are necessary

[2] O. Afisiadis, A. Balatsoukas-Stimming and A. Burg, ”A low-complexity improved successive cancellation decoder for polar codes,” 2014 48th Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, 2014, pp. 2116-2120.

Example: $N = 16$, $K + C = 12$, $T_{max} = 4$

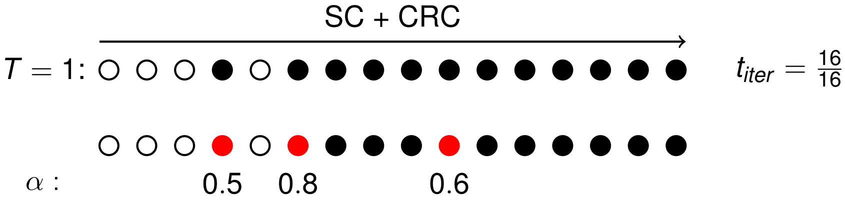
- First iteration: pass/fail?

$\xrightarrow{\text{SC + CRC}}$

$T = 1$: ○ ○ ○ ● ○ ● ● ● ● ● ● ● ● ● ● ● $t_{iter} = \frac{16}{16}$

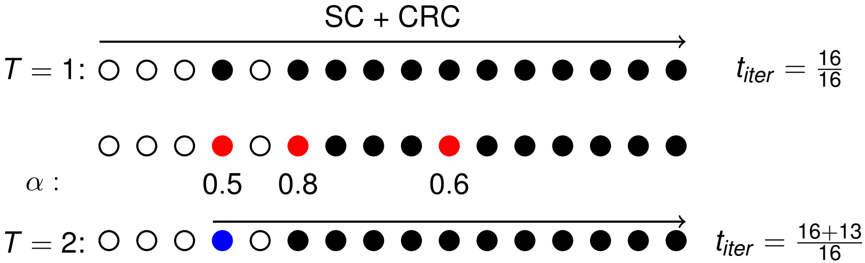
Example: $N = 16$, $K + C = 12$, $T_{max} = 4$

- Fail: Flip $T_{max} - 1$ least reliable indices, one at a time



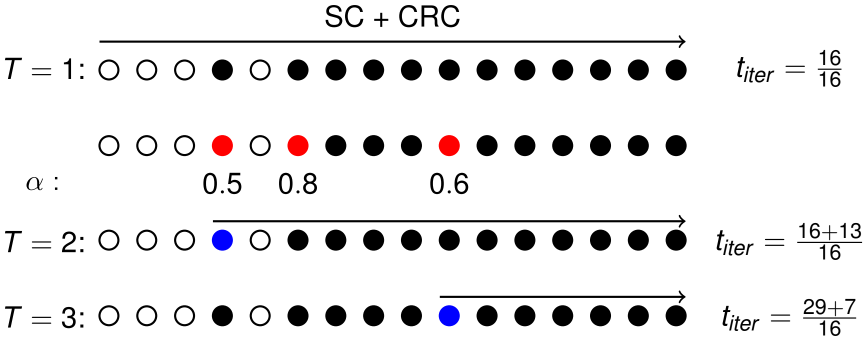
Example: $N = 16$, $K + C = 12$, $T_{max} = 4$

- $T = 2$: Flip least reliable index



Example: $N = 16$, $K + C = 12$, $T_{max} = 4$

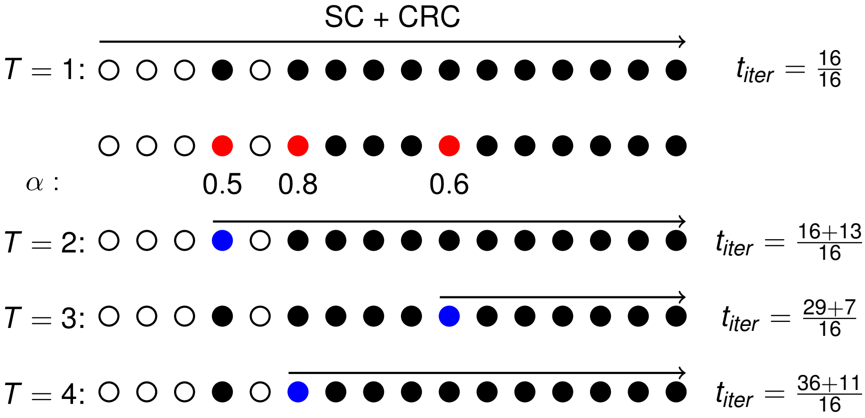
- $T = 3$: Flip second least reliable index



SC + CRC
$T = 1$:
$t_{iter} = \frac{16}{16}$
α :
0.5
0.8
0.6
$T = 2$:
$t_{iter} = \frac{16+13}{16}$
$T = 3$:
$t_{iter} = \frac{29+7}{16}$

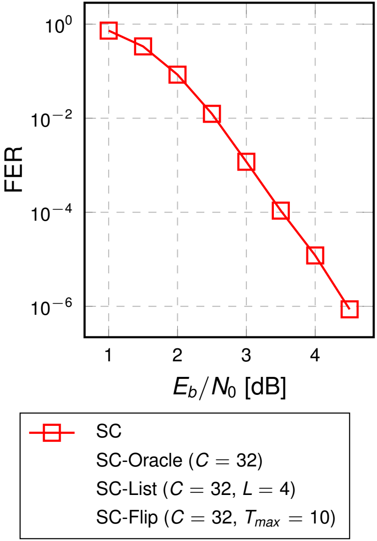
Example: $N = 16$, $K + C = 12$, $T_{max} = 4$

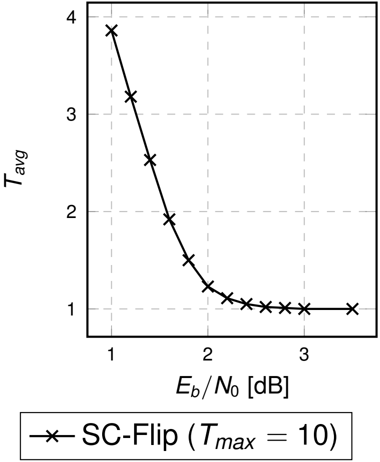
- $T = 4$: Flip third least reliable index


SC + CRC
$T = 1$:
$t_{iter} = \frac{16}{16}$
α :
0.5
0.8
0.6
$T = 2$:
$t_{iter} = \frac{16+13}{16}$
$T = 3$:
$t_{iter} = \frac{29+7}{16}$
$T = 4$:
$t_{iter} = \frac{36+11}{16}$

SC-Flip Decoding: Insights

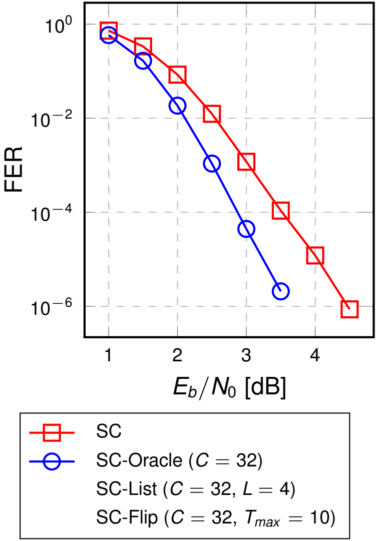
- $PC(1024, 512)$

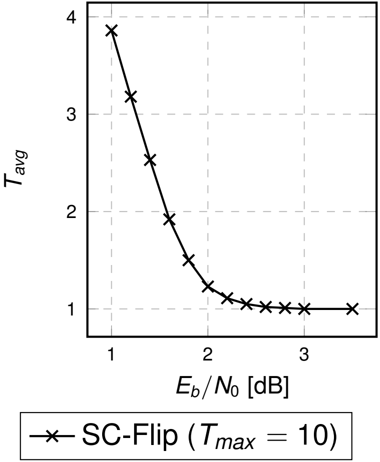




SC-Flip Decoding: Insights

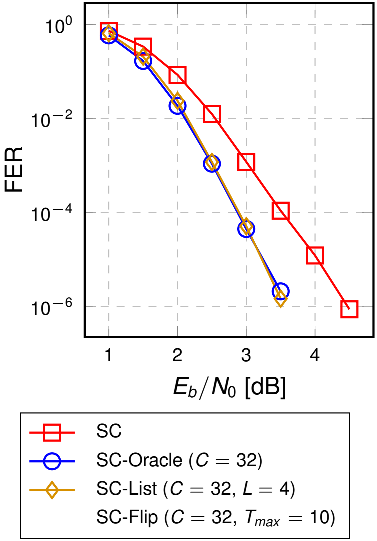
- $PC(1024, 512)$

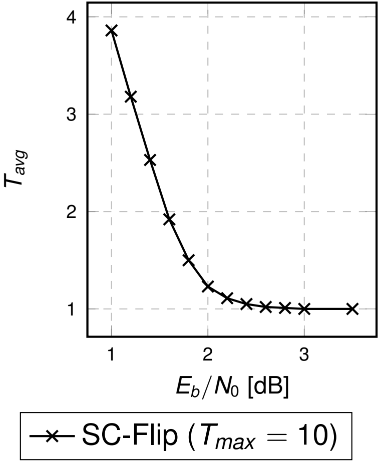




SC-Flip Decoding: Insights

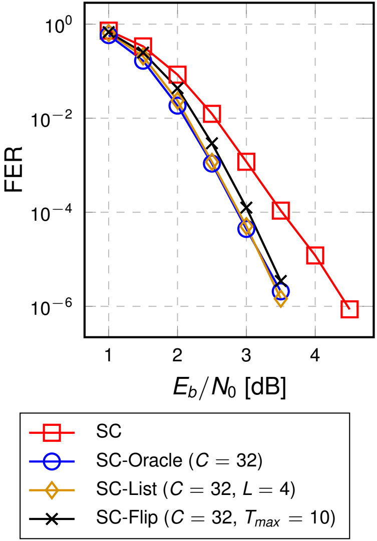
- $PC(1024, 512)$

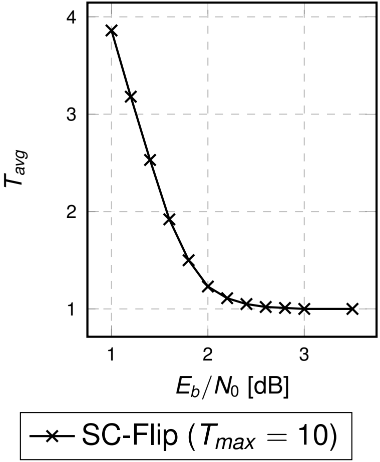




SC-Flip Decoding: Insights

- $PC(1024, 512)$





Partitioned SC-Flip Decoding Algorithm (PSCF)

Observations:

Idea:

Partitioned SC-Flip Decoding Algorithm (PSCF)

Observations:

- ✗ SC-Flip does not achieve SC-Oracle performance within a practical T_{max}

Idea:

Partitioned SC-Flip Decoding Algorithm (PSCF)

Observations:

- ✗ SC-Flip does not achieve SC-Oracle performance within a practical T_{max}
- ✗ The search space for the flipping index is too large

Idea:

Partitioned SC-Flip Decoding Algorithm (PSCF)

Observations:

- ✗ SC-Flip does not achieve SC-Oracle performance within a practical T_{max}
- ✗ The search space for the flipping index is too large

Idea:

- ► Divide the codeword into a number of partitions
  - ► Each partition is protected by its own CRC

Partitioned SC-Flip Decoding Algorithm (PSCF)

Observations:

- ✗ SC-Flip does not achieve SC-Oracle performance within a practical T_{max}
- ✗ The search space for the flipping index is too large

Idea:

- ▸ Divide the codeword into a number of partitions
  - ▸ Each partition is protected by its own CRC
- ✓ The search space for the flipping index is narrowed
  - ✓ **Reduced iterations, enables early termination**

Partitioned SC-Flip Decoding Algorithm (PSCF)

Observations:

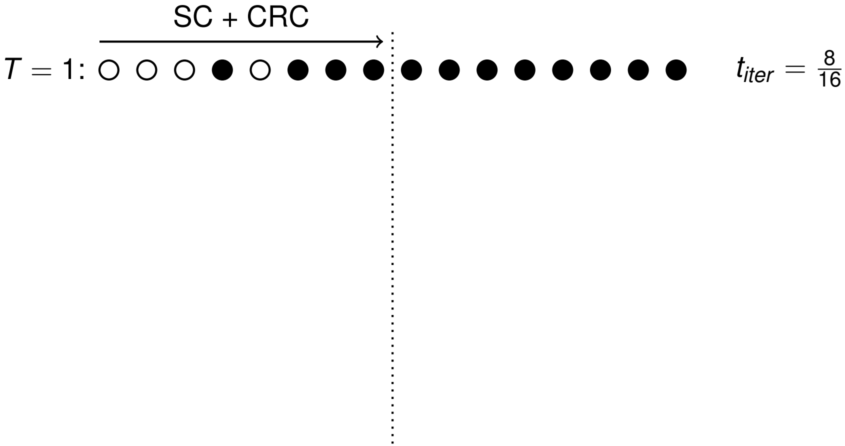
- ✗ SC-Flip does not achieve SC-Oracle performance within a practical T_{max}
- ✗ The search space for the flipping index is too large

Idea:

- ▸ Divide the codeword into a number of partitions
  - ▸ Each partition is protected by its own CRC
- ✓ The search space for the flipping index is narrowed
  - ✓ **Reduced iterations, enables early termination**
- ✓ Possibility of correcting **multiple channel-induced errors**
  - ! ...if each error resides in a different partition

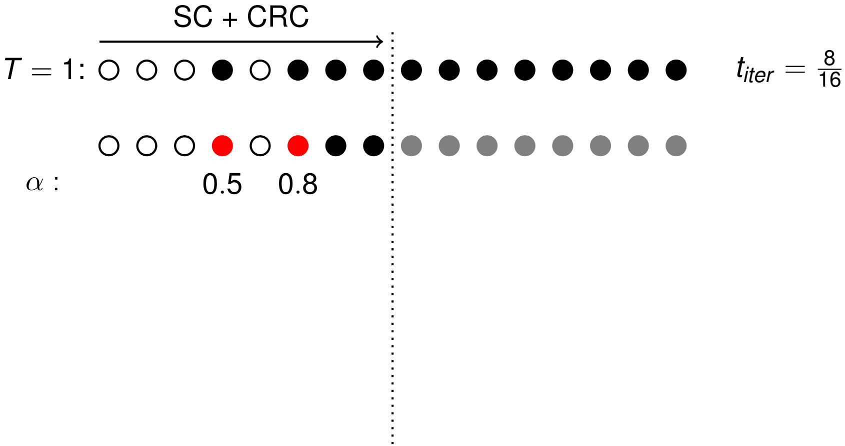
PSCF Decoding Algorithm

- Codeword is divided into sub-blocks


SC + CRC
$T = 1$: ○ ○ ○ ● ○ ● ● ● ● ● ● ● ● ● ● ●
$t_{iter} = \frac{8}{16}$

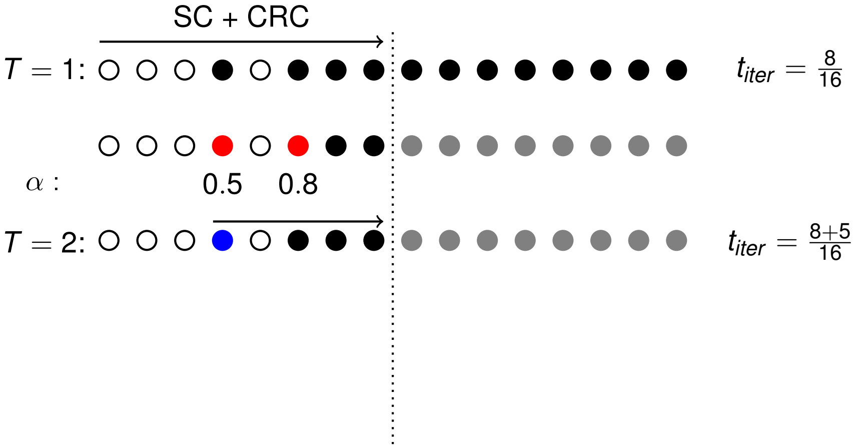
PSCF Decoding Algorithm

- SC-Flip is applied to each sub-block independently



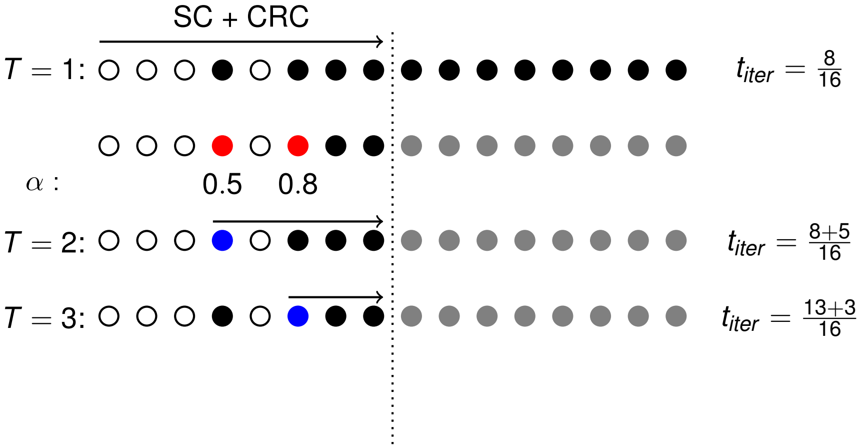
PSCF Decoding Algorithm

- SC-Flip is applied to each sub-block independently



PSCF Decoding Algorithm

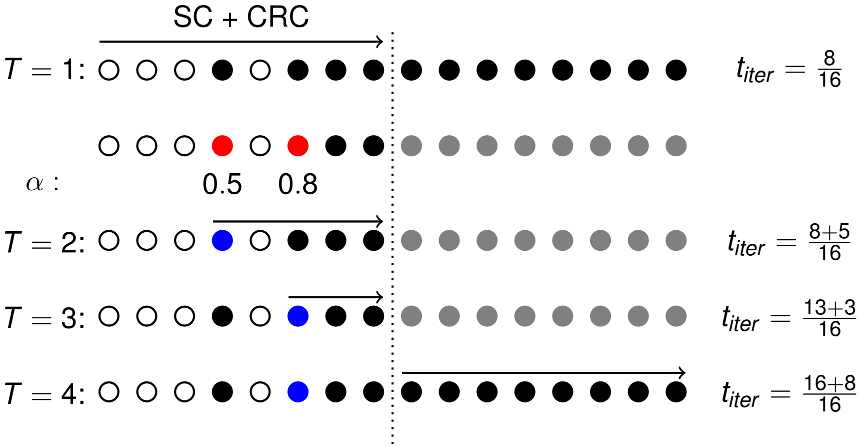
- SC-Flip is applied to each sub-block independently



SC + CRC
$T = 1$:
$t_{iter} = \frac{8}{16}$
α :
0.5
0.8
$T = 2$:
$t_{iter} = \frac{8+5}{16}$
$T = 3$:
$t_{iter} = \frac{13+3}{16}$

PSCF Decoding Algorithm

- Shorter iterations result reduced T_{avg}



SC + CRC

$T = 1$: $t_{iter} = \frac{8}{16}$

α : 0.5 0.8

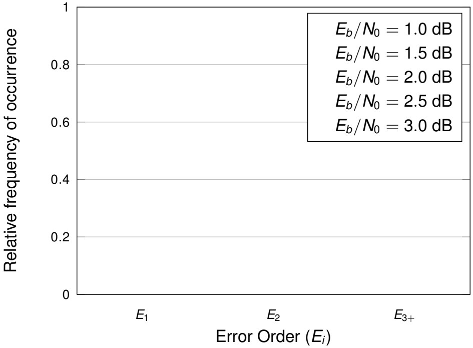
$T = 2$: $t_{iter} = \frac{8+5}{16}$

$T = 3$: $t_{iter} = \frac{13+3}{16}$

$T = 4$: $t_{iter} = \frac{16+8}{16}$

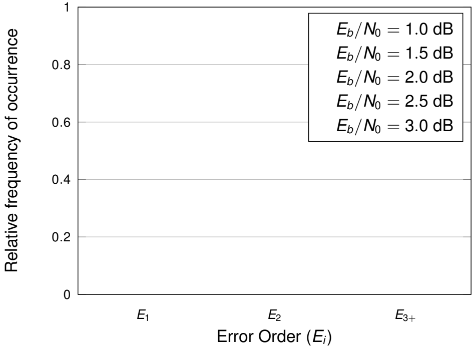
How to Partition The Codeword

- Objective: to cover equal amount of error probability in each partition



How to Partition The Codeword

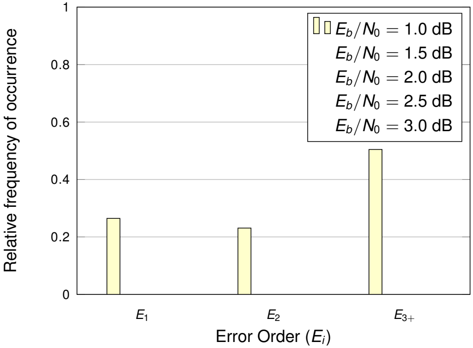
- Objective: to cover equal amount of error probability in each partition



- Each partition should cover an equal probability of error occurrence

How to Partition The Codeword

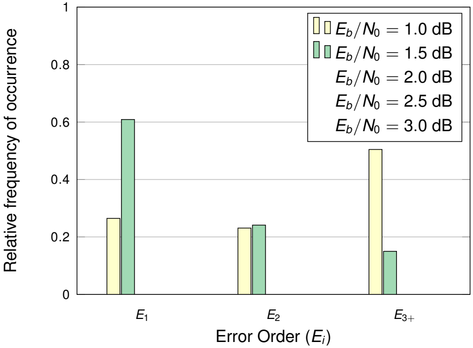
- Objective: to cover equal amount of error probability in each partition



- Each partition should cover an equal probability of error occurrence

How to Partition The Codeword

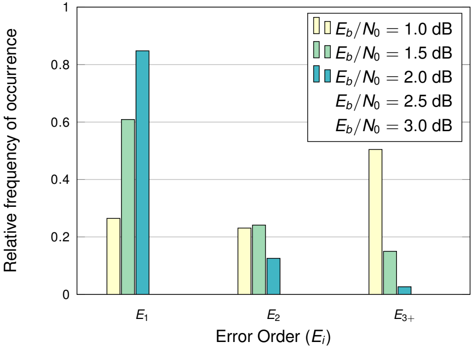
- Objective: to cover equal amount of error probability in each partition



- Each partition should cover an equal probability of error occurrence

How to Partition The Codeword

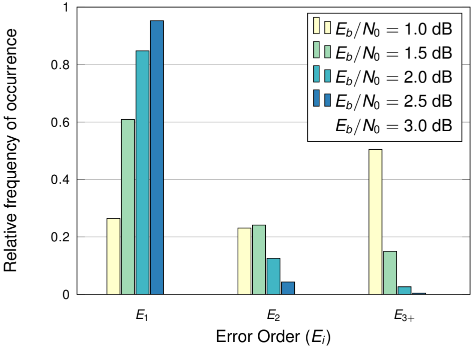
- Objective: to cover equal amount of error probability in each partition



- Each partition should cover an equal probability of error occurrence

How to Partition The Codeword

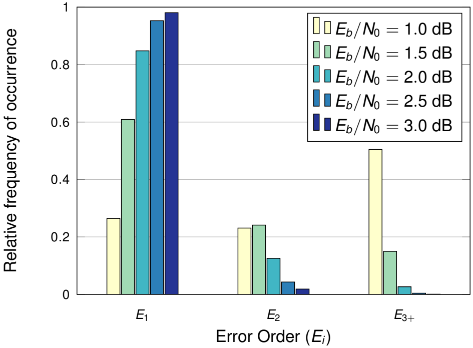
- Objective: to cover equal amount of error probability in each partition



- Each partition should cover an equal probability of error occurrence

How to Partition The Codeword

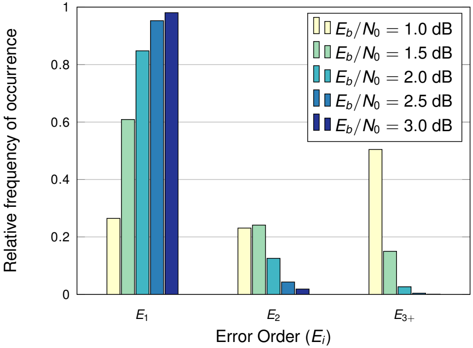
- Objective: to cover equal amount of error probability in each partition



- Each partition should cover an equal probability of error occurrence

How to Partition The Codeword

- Objective: to cover equal amount of error probability in each partition

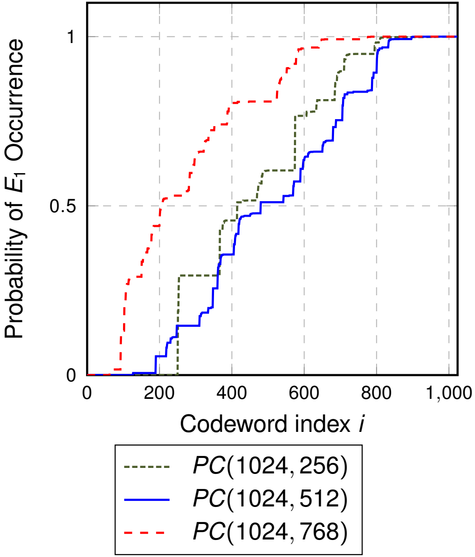


- Each partition should cover an equal probability of error occurrence

- E_1 dominates the error occurrence at medium/high SNR → can be used to approximate partitioning

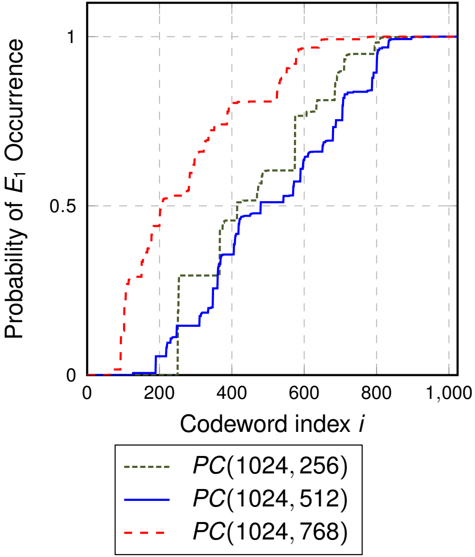
Codeword Partitioning

- Figure: CDF based on E_1 ($E_b/N_0 = 2.5$ dB)



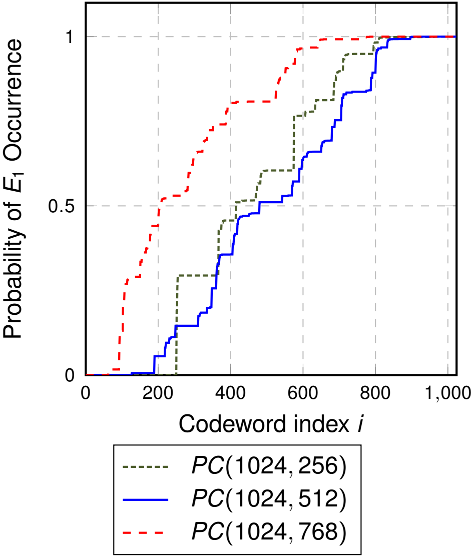
Codeword Partitioning

- Figure: CDF based on E_1 ($E_b/N_0 = 2.5$ dB)
- Objective: To find partitioning indices $\boldsymbol{\rho}$ such that $0 < \boldsymbol{\rho} < N$



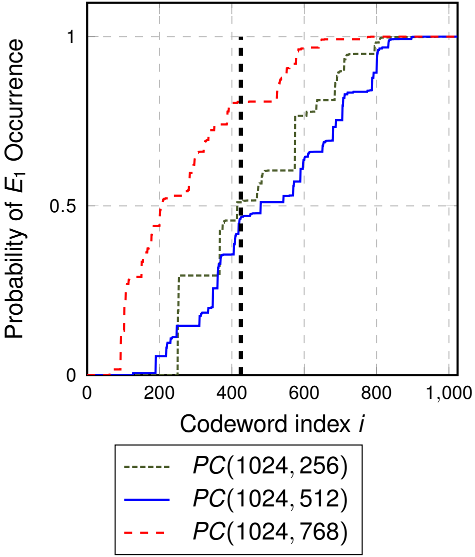
Codeword Partitioning

- Figure: CDF based on E_1 ($E_b/N_0 = 2.5$ dB)
- Objective: To find partitioning indices ρ such that $0 < \rho < N$
- Example: $P = 2$



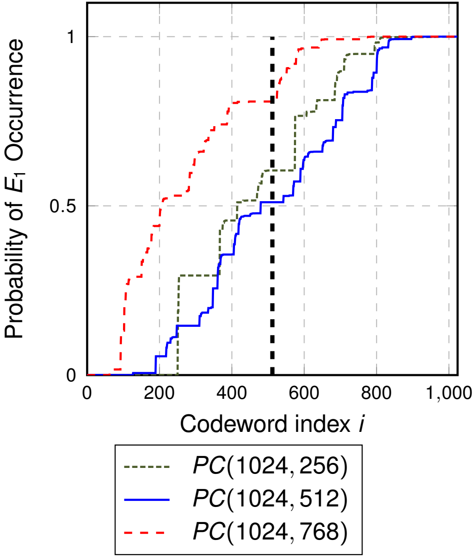
Codeword Partitioning

- Figure: CDF based on E_1 ($E_b/N_0 = 2.5$ dB)
- Objective: To find partitioning indices $\boldsymbol{\rho}$ such that $0 < \boldsymbol{\rho} < N$
- Example: $P = 2$
  - $\rho_{(1024,256)} = 425$



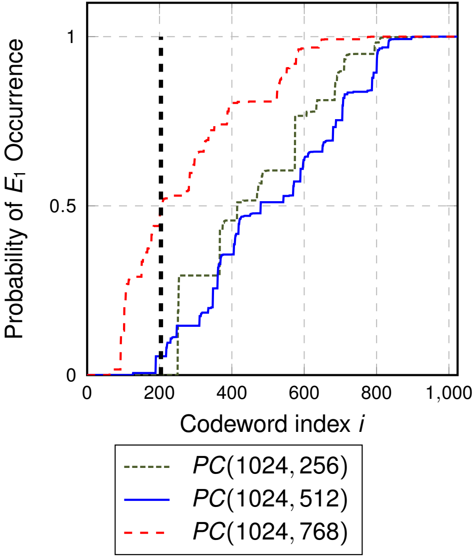
Codeword Partitioning

- Figure: CDF based on E_1 ($E_b/N_0 = 2.5$ dB)
- Objective: To find partitioning indices $\boldsymbol{\rho}$ such that $0 < \boldsymbol{\rho} < N$
- Example: $P = 2$
  - $\rho_{(1024,256)} = 425$
  - $\rho_{(1024,512)} = 512$



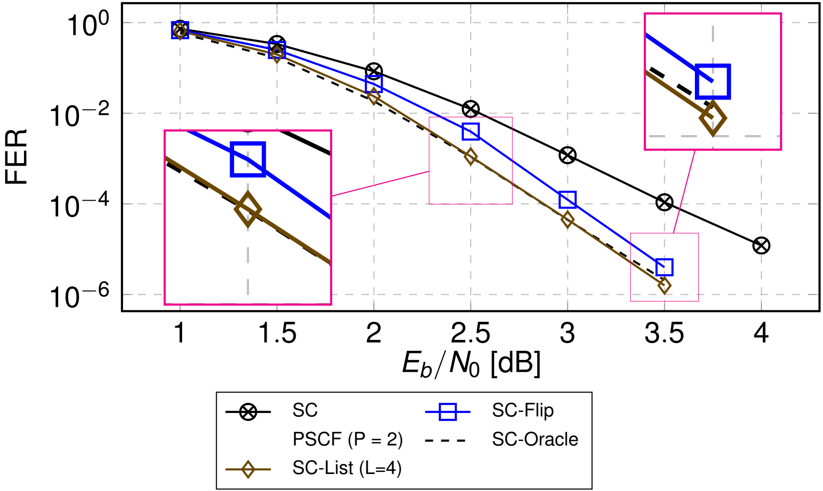
Codeword Partitioning

- Figure: CDF based on E_1 ($E_b/N_0 = 2.5$ dB)
- Objective: To find partitioning indices $\boldsymbol{\rho}$ such that $0 < \boldsymbol{\rho} < N$
- Example: $P = 2$
  - $\rho_{\mathbf{(1024,256)}} = 425$
  - $\rho_{\mathbf{(1024,512)}} = 512$
  - $\rho_{\mathbf{(1024,768)}} = 204$



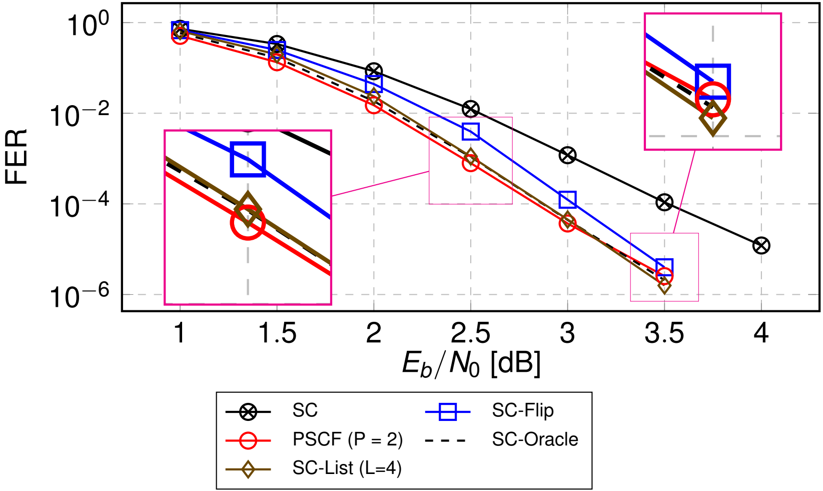
Simulation Results - Performance

- $PC(1024, 512)$, $C = 32$, $T_{max} = 10$, $P = 2$



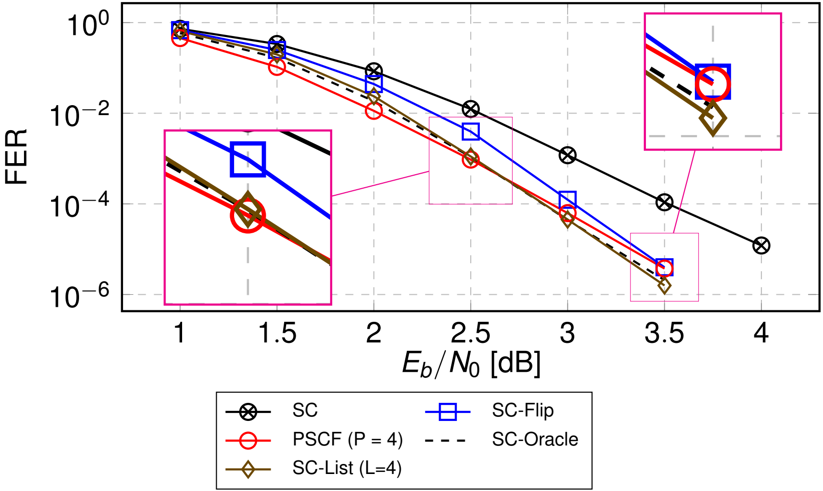
Simulation Results - Performance

- $PC(1024, 512)$, $C = 32$, $T_{max} = 10$, $P = 2$



Simulation Results - Performance (cont'd)

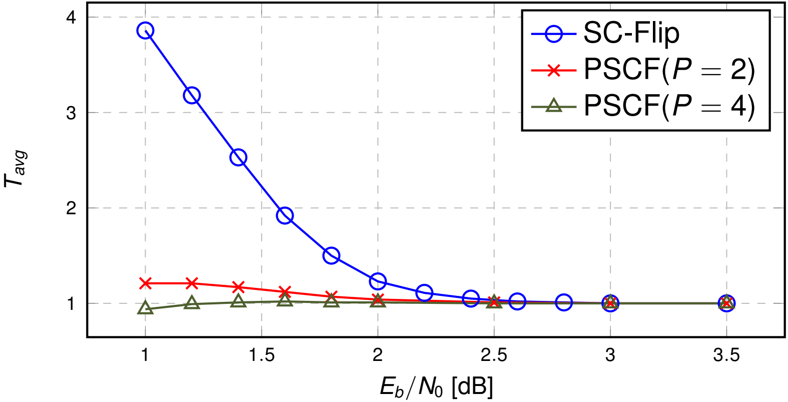
- $PC(1024, 512)$, $C = 32$, $T_{max} = 10$, $P = 4$



Simulation Results - Average Iterations

- $PC(1024, 512)$, $C = 32$

- At matching FER with SC-Flip ($T_{max} = 10$)



Conclusion

We have presented Partitioned SC-Flip decoding algorithm for polar codes

Conclusion

We have presented Partitioned SC-Flip decoding algorithm for polar codes

- Correcting more than a single error is possible via partitioning

Conclusion

We have presented Partitioned SC-Flip decoding algorithm for polar codes

- Correcting more than a single error is possible via partitioning
  - ✓ Improved error-correction performance by up to 0.26 dB with $P = 2$ compared to SC-Flip

Conclusion

We have presented Partitioned SC-Flip decoding algorithm for polar codes

- Correcting more than a single error is possible via partitioning
    - ✓ Improved error-correction performance by up to 0.26 dB with $P = 2$ compared to SC-Flip
- Significantly reduced T_{ave} compared to baseline SC-Flip

Conclusion

We have presented Partitioned SC-Flip decoding algorithm for polar codes

- Correcting more than a single error is possible via partitioning
  - ✓ Improved error-correction performance by up to 0.26 dB with $P = 2$ compared to SC-Flip
- Significantly reduced T_{ave} compared to baseline SC-Flip
  - ✓ Reduction by up to 3.2× with $P = 2$ and 4.1× with $P = 4$

Conclusion

We have presented Partitioned SC-Flip decoding algorithm for polar codes

- Correcting more than a single error is possible via partitioning
  - ✓ Improved error-correction performance by up to 0.26 dB with $P = 2$ compared to SC-Flip
- Significantly reduced T_{ave} compared to baseline SC-Flip
  - ✓ Reduction by up to $3.2\times$ with $P = 2$ and $4.1\times$ with $P = 4$
  - ✓ Leads to **increased average throughput** and **reduced energy consumption**

Conclusion

We have presented Partitioned SC-Flip decoding algorithm for polar codes

- Correcting more than a single error is possible via partitioning
  - ✓ Improved error-correction performance by up to 0.26 dB with $P = 2$ compared to SC-Flip
- Significantly reduced T_{ave} compared to baseline SC-Flip
  - ✓ Reduction by up to $3.2\times$ with $P = 2$ and $4.1\times$ with $P = 4$
  - ✓ Leads to **increased average throughput** and **reduced energy consumption**
- Cummulative error distribution schemes help decide better partitioning

Thank you!