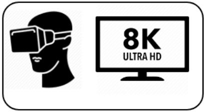
# Simplified Dynamic SC-Flip Polar Decoding

**Furkan Ercan***, Thibaud Tonnellier, Nghia Doan, Warren J. Gross
Integrated Systems for Information Processing (ISIP) Lab
McGill University
Montréal, Québec, Canada

May 8, 2020

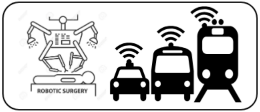
# 5G Use Cases

Enhanced Mobile Broadband (**eMBB**)




**High Throughput**

Ultra-Reliable Low-Latency Communications (**URLLC**)




**Low Latency**
**High Reliability**

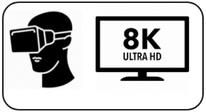
Massive Machine-Type Communications (**mMTC**)

**Massive Connectivity**
**Energy Efficiency**

- 5G prioritizes various targets based on the use case.

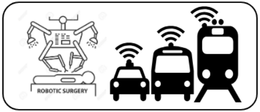
# 5G Use Cases

Enhanced Mobile Broadband (**eMBB**)




**High Throughput**

Ultra-Reliable Low-Latency Communications (**URLLC**)




**Low Latency**
**High Reliability**

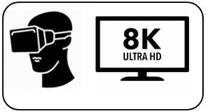
Massive Machine-Type Communications (**mMTC**)

**Massive Connectivity**
**Energy Efficiency**

- 5G prioritizes various targets based on the use case.
- Polar codes provably achieve channel capacity.
- They are involved in 5G eMBB control channel.

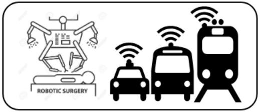
# 5G Use Cases

Enhanced Mobile Broadband (**eMBB**)




**High Throughput**

Ultra-Reliable Low-Latency Communications (**URLLC**)




**Low Latency**
**High Reliability**

Massive Machine-Type Communications (**mMTC**)

**Massive Connectivity**
**Energy Efficiency**

- 5G prioritizes various targets based on the use case.
- Polar codes provably achieve channel capacity.
- They are involved in 5G eMBB control channel.
- Currently, polar codes are being evaluated for other use cases.

# An Overview of SC Algorithms

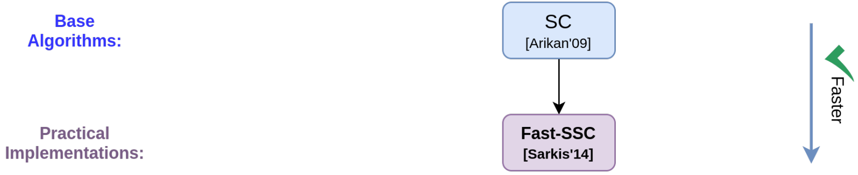
**Base Algorithms:**

**SC**
**[Arikan'09]**

Successive Cancellation (SC) Decoding

- ✓ Simple encoding/decoding
- ✗ Mediocre performance at practical lengths
- ✗ Sequential, long latency

# An Overview of SC Algorithms

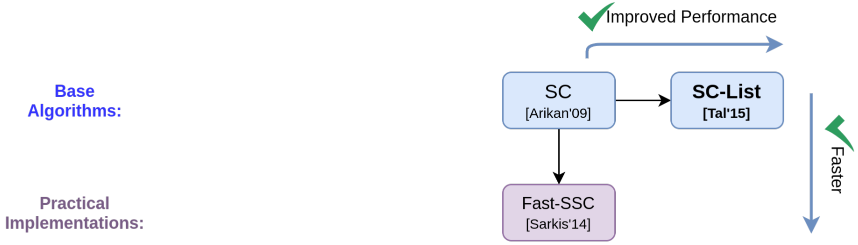



Fast-SSC Decoding

✓ $\approx 10\times$ less latency

- No error correction performance degradation

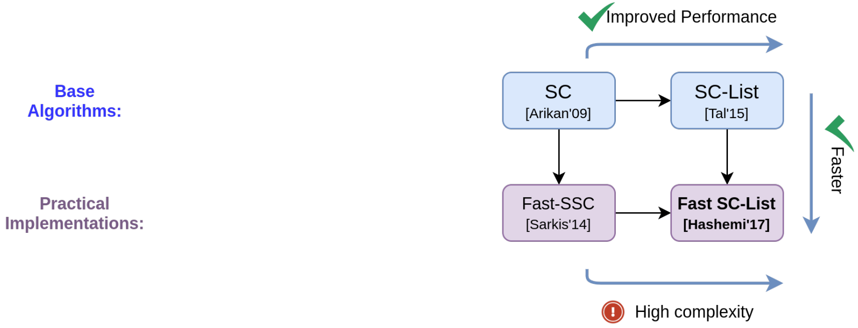
# An Overview of SC Algorithms




SC-List (SCL) Decoding

- ✓ Improved performance
- ✗ Increased complexity

# An Overview of SC Algorithms

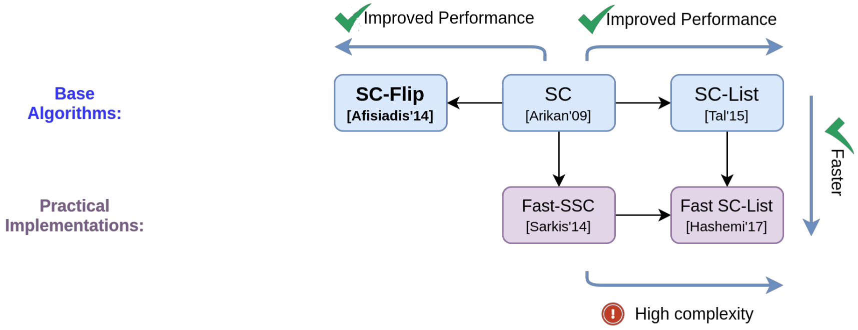



SC-List (SCL) Decoding

✓ Improved performance

✗ Increased complexity

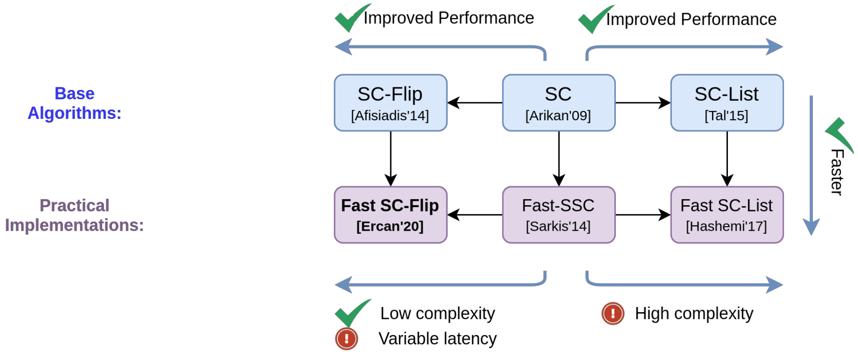
# An Overview of SC Algorithms




SC-Flip (SCF) Decoding

- ✓ Some improved performance
- ✓ Low complexity
- ✗ Variable latency

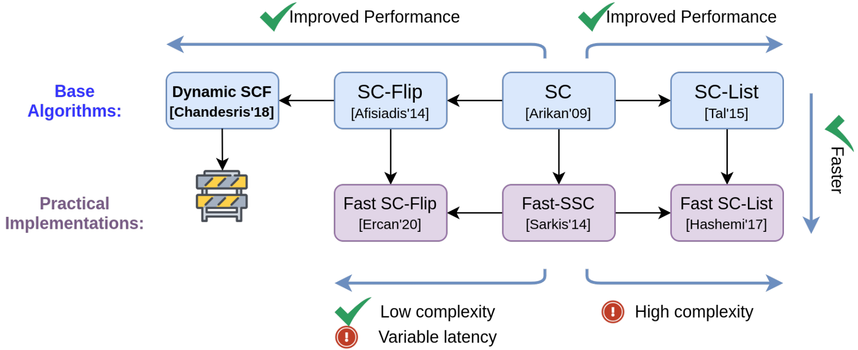
# An Overview of SC Algorithms




SC-Flip (SCF) Decoding

- ✓ Some improved performance
- ✓ Low complexity
- ✗ Variable latency

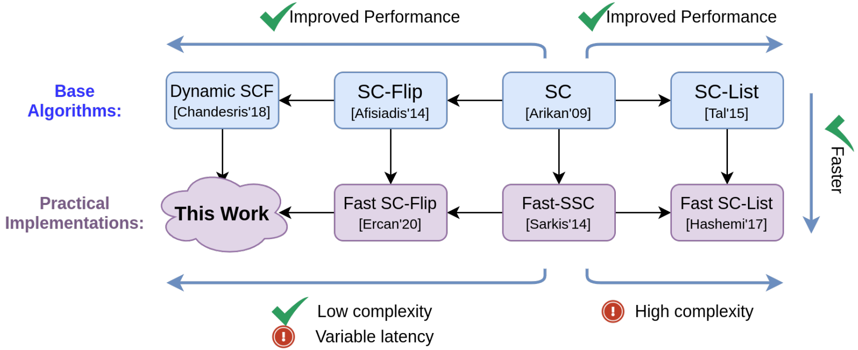
# An Overview of SC Algorithms




Dynamic SCF (DSCF) Decoding

- ✓ Better improved performance
- ✗ Expensive computations (log, exp, $\times$)
- ✗ No practical implementation

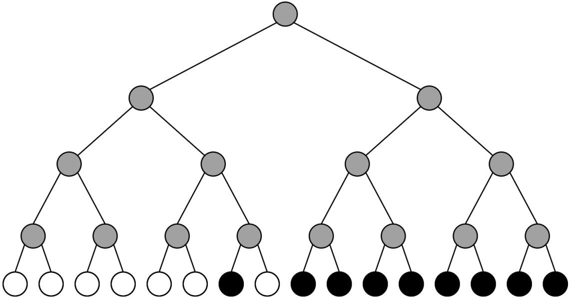
# An Overview of SC Algorithms



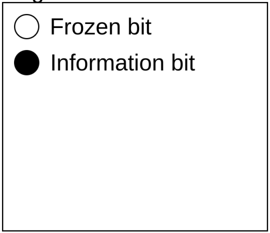

This Work

- ✓ No expensive computations
- ✓ Introduce fast decoding techniques
- ✓ First steps towards practical implementations

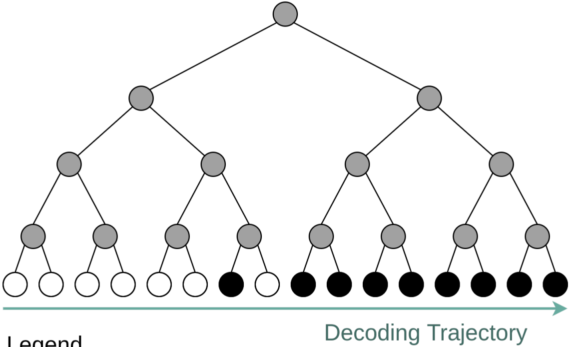
# SC-Flip (SCF) Decoding



Legend



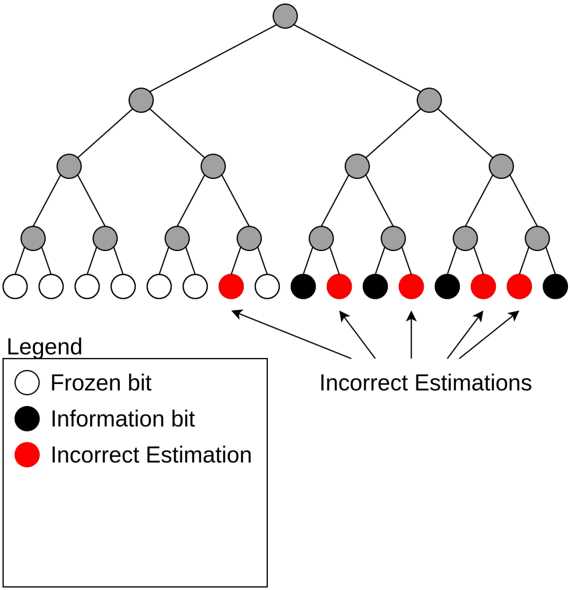
# SC-Flip (SCF) Decoding



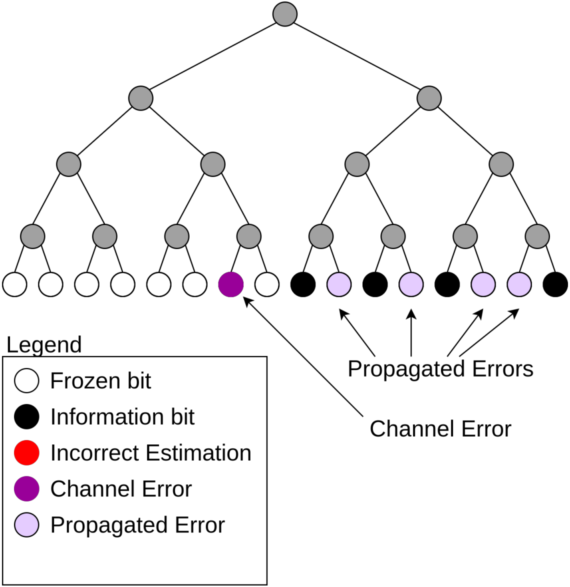

Legend

- ○ Frozen bit
- ● Information bit

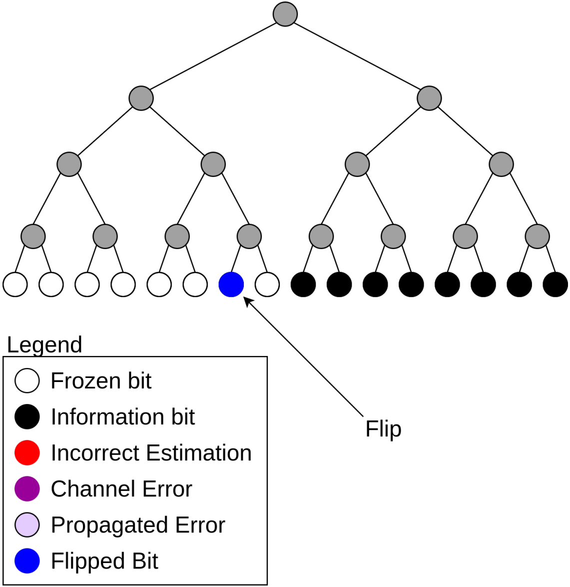
# SC-Flip (SCF) Decoding



# SC-Flip (SCF) Decoding



# SC-Flip (SCF) Decoding



# Problems with SCF Algorithm

- Metric for SCF for node index i: $|L_i|$ where $L$ is LLR.

# Problems with SCF Algorithm

- Metric for SCF for node index i: $|L_i|$ where $L$ is LLR.
- Performance improvement of SCF is limited:
  - Comparable to SCL with small list sizes.

# Problems with SCF Algorithm

- Metric for SCF for node index i: $|L_i|$ where $L$ is LLR.
- Performance improvement of SCF is limited:
  - Comparable to SCL with small list sizes.
- Two main problems of SCF:
  - Metric cannot distinguish channel errors from propagated errors.
  - Only one error can be corrected.

# Dynamic SC-Flip (DSCF) Decoding

Dynamic SC-Flip (DSCF) algorithm tackles both issues of SCF decoding:

- A better metric that can distinguish channel errors from propagated errors.

# Dynamic SC-Flip (DSCF) Decoding

Dynamic SC-Flip (DSCF) algorithm tackles both issues of SCF decoding:

- A better metric that can distinguish channel errors from propagated errors.
- This gives an opportunity to tackle more than one channel error.

# Dynamic SC-Flip (DSCF) Decoding

Dynamic SC-Flip (DSCF) algorithm tackles both issues of SCF decoding:

- A better metric that can distinguish channel errors from propagated errors.
- This gives an opportunity to tackle more than one channel error.
- Therefore, performance is improved greatly.

# Dynamic SC-Flip (DSCF) Decoding

- Let $\omega$ denote the *decoding order*.
- Let $\mathcal{E}_\omega = \{i_1, \ldots, i_\omega\}$ denote the set of bit-flipping indices.
- Note that the set $\mathcal{E}_\omega$ is built *progressively* over $\mathcal{E}_{\omega-1}$.

# Dynamic SC-Flip (DSCF) Decoding

- Let $\omega$ denote the *decoding order*.
- Let $\mathcal{E}_\omega = \{i_1, \ldots, i_\omega\}$ denote the set of bit-flipping indices.
- Note that the set $\mathcal{E}_\omega$ is built *progressively* over $\mathcal{E}_{\omega-1}$.

The metric for DSCF decoding, based on LLRs, is computed as

$$M_\alpha(\mathcal{E}_\omega) = \sum_{j \in \mathcal{E}_\omega} |L^0[\mathcal{E}_{\omega-1}]_j| + S_\alpha(\mathcal{E}_\omega)$$

where

$$S_\alpha(\mathcal{E}_\omega) = \frac{1}{\alpha} \sum_{\substack{j \leqslant i_\omega \\ \forall j \in \mathcal{A}}} \log(1 + \exp(-\alpha |L^0[\mathcal{E}_{\omega-1}]_j|))$$

# Dynamic SC-Flip (DSCF) Decoding

- Let $\omega$ denote the *decoding order*.
- Let $\mathcal{E}_\omega = \{i_1, \dots, i_\omega\}$ denote the set of bit-flipping indices.
- Note that the set $\mathcal{E}_\omega$ is built *progressively* over $\mathcal{E}_{\omega-1}$.

The metric for DSCF decoding, based on LLRs, is computed as

LLR magnitudes at flipping indices

$$M_\alpha(\mathcal{E}_\omega) = \underbrace{\sum_{j \in \mathcal{E}_\omega} |L^0[\mathcal{E}_{\omega-1}]_j|}_{} + S_\alpha(\mathcal{E}_\omega)$$

where

$$S_\alpha(\mathcal{E}_\omega) = \frac{1}{\alpha} \sum_{\substack{j \leqslant i_\omega \\ \forall j \in \mathcal{A}}} \log(1 + \exp(-\alpha |L^0[\mathcal{E}_{\omega-1}]_j|))$$

# Dynamic SC-Flip (DSCF) Decoding

- Let $\omega$ denote the *decoding order*.
- Let $\mathcal{E}_\omega = \{i_1, \ldots, i_\omega\}$ denote the set of bit-flipping indices.
- Note that the set $\mathcal{E}_\omega$ is built *progressively* over $\mathcal{E}_{\omega-1}$.

The metric for DSCF decoding, based on LLRs, is computed as

$$M_\alpha(\mathcal{E}_\omega) = \underbrace{\sum_{j \in \mathcal{E}_\omega} |L^0[\mathcal{E}_{\omega-1}]_j|}_{\text{LLR magnitudes at flipping indices}} + \underbrace{S_\alpha(\mathcal{E}_\omega)}_{\text{Part that distinguishes channel errors from others}}$$

where

$$S_\alpha(\mathcal{E}_\omega) = \frac{1}{\alpha} \sum_{\substack{j \leqslant i_\omega \\ \forall j \in \mathcal{A}}} \log(1 + \exp(-\alpha |L^0[\mathcal{E}_{\omega-1}]_j|))$$

# Dynamic SC-Flip (DSCF) Decoding

- Let $\omega$ denote the *decoding order*.
- Let $\mathcal{E}_\omega = \{i_1, \ldots, i_\omega\}$ denote the set of bit-flipping indices.
- Note that the set $\mathcal{E}_\omega$ is built *progressively* over $\mathcal{E}_{\omega-1}$.

The metric for DSCF decoding, based on LLRs, is computed as

$$M_\alpha(\mathcal{E}_\omega) = \underbrace{\sum_{j \in \mathcal{E}_\omega} |L^0[\mathcal{E}_{\omega-1}]_j|}_{\text{LLR magnitudes at flipping indices}} + \underbrace{S_\alpha(\mathcal{E}_\omega)}_{\text{Part that distinguishes channel errors from others}}$$

where

$$S_\alpha(\mathcal{E}_\omega) = \frac{1}{\underbrace{\alpha}_{\text{positive constant}}} \sum_{\substack{j \leqslant i_\omega \\ \forall j \in \mathcal{A}}} \log(1 + \exp(-\alpha |L^0[\mathcal{E}_{\omega-1}]_j|))$$

# Dynamic SC-Flip (DSCF) Decoding

- Let $\omega$ denote the *decoding order*.
- Let $\mathcal{E}_\omega = \{i_1, \ldots, i_\omega\}$ denote the set of bit-flipping indices.
- Note that the set $\mathcal{E}_\omega$ is built *progressively* over $\mathcal{E}_{\omega-1}$.

The metric for DSCF decoding, based on LLRs, is computed as

LLR magnitudes at flipping indices

Part that distinguishes channel errors from others

So we can't ignore it

$$M_\alpha(\mathcal{E}_\omega) = \sum_{j\in\mathcal{E}_\omega} |L^0[\mathcal{E}_{\omega-1}]_j| + S_\alpha(\mathcal{E}_\omega)$$

where

$$S_\alpha(\mathcal{E}_\omega) = \frac{1}{\alpha}\sum_{\substack{j\leqslant i_\omega \\ \forall j\in\mathcal{A}}} \log(1+\exp(-\alpha|L^0[\mathcal{E}_{\omega-1}]_j|))$$

positive constant

Based only on LLRs

# Simplifications over the DSCF Metric

First, we reformulate $S_\alpha(\mathcal{E}_\omega)$ as

$$S_\alpha(\mathcal{E}_\omega) = \sum_{\substack{j \leqslant i_\omega \\ \forall j \in \mathcal{A}}} f_\alpha(|L^0[\mathcal{E}_{\omega-1}]_j|),$$

where

$$f_\alpha(x) = \frac{1}{\alpha} \log(1 + \exp(-\alpha x)).$$

# Simplifications over the DSCF Metric

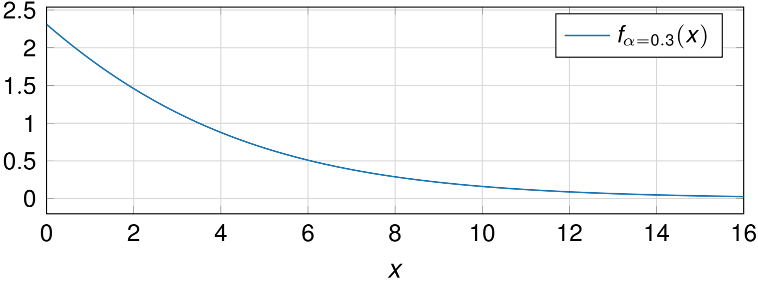
First, we reformulate $S_\alpha(\mathcal{E}_\omega)$ as

$$S_\alpha(\mathcal{E}_\omega) = \sum_{\substack{j \leqslant i_\omega \\ \forall j \in \mathcal{A}}} f_\alpha(|L^0[\mathcal{E}_{\omega-1}]_j|),$$

where

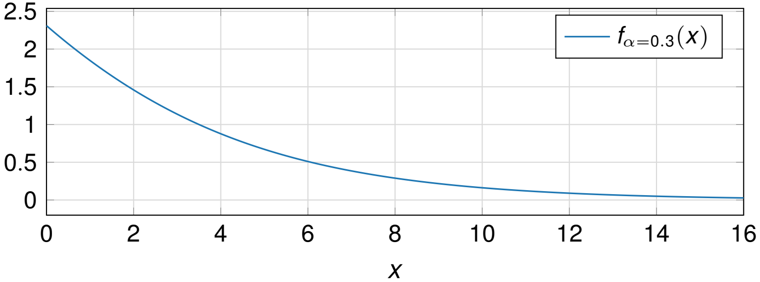
$$f_\alpha(x) = \frac{1}{\alpha} \log(1 + \exp(-\alpha x)).$$

Then, we take a closer look at the behavior of $f_\alpha(x)$. Following the original DSCF algorithm, $\alpha = .3$.



# Simplifications over the DSCF Metric

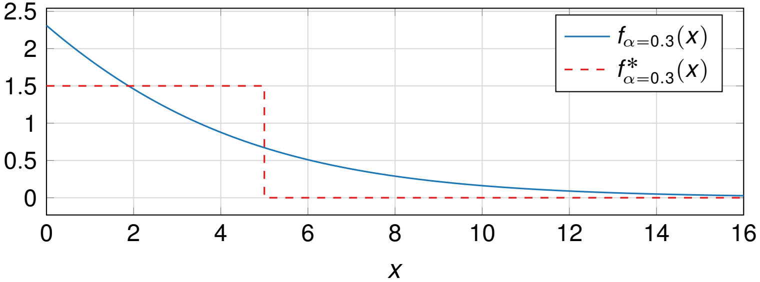
- Interestingly, a similar problem was encountered for Turbo codes in 1998*.




* W. Gross and P. Gulak. "Simplified MAP algorithm suitable for implementation of turbo decoders," Electronics Letters, vol. 34, no. 16, pp. 15771578, Aug 1998.

# Simplifications over the DSCF Metric

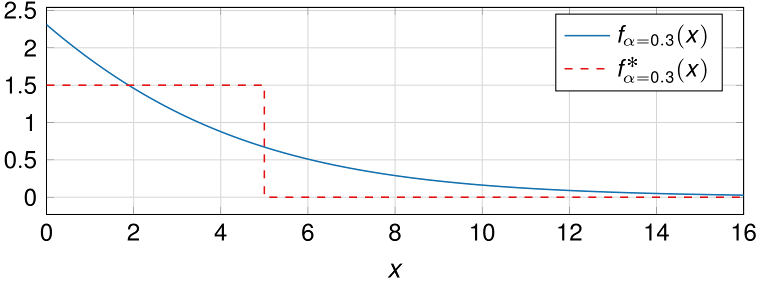
- Interestingly, a similar problem was encountered for Turbo codes in 1998*.




* W. Gross and P. Gulak. "Simplified MAP algorithm suitable for implementation of turbo decoders," Electronics Letters, vol. 34, no. 16, pp. 15771578, Aug 1998.

# Simplifications over the DSCF Metric

- Interestingly, a similar problem was encountered for Turbo codes in 1998*.




We replace $f_{\alpha=0.3}(x)$ with

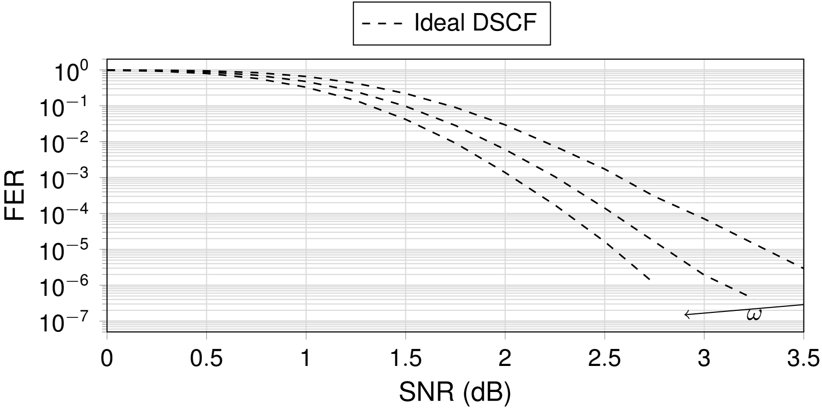
$$f^*_{\alpha=0.3}(x) = \begin{cases} \frac{3}{2}, & \text{if } |x| \leqslant 5 \\ 0, & \text{otherwise.} \end{cases}$$

The values in $f^*_{\alpha=0.3}(x)$ are chosen in favor of quantization.

* W. Gross and P. Gulak. "Simplified MAP algorithm suitable for implementation of turbo decoders," Electronics Letters, vol. 34, no. 16, pp. 15771578, Aug 1998.

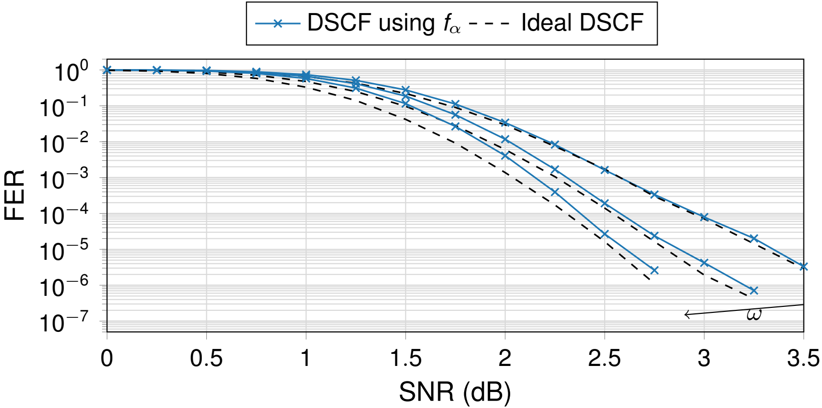
# Impact of the Simplification on Performance

- $PC(N, K) = PC(1024, 512)$, CRC length $C = 16$ bits.
- $T_{\max} \in \{10, 40, 200\}$ for $\omega \in \{1, 2, 3\}$.



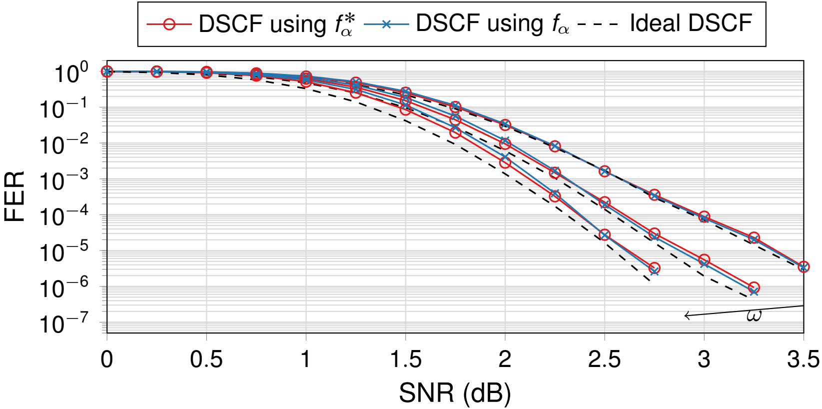
# Impact of the Simplification on Performance

- $PC(N, K) = PC(1024, 512)$, CRC length $C = 16$ bits.
- $T_{\max} \in \{10, 40, 200\}$ for $\omega \in \{1, 2, 3\}$.



# Impact of the Simplification on Performance

- $PC(N, K) = PC(1024, 512)$, CRC length $C = 16$ bits.
- $T_{\text{max}} \in \{10, 40, 200\}$ for $\omega \in \{1, 2, 3\}$.

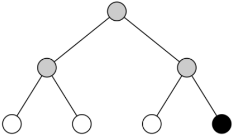


# Incorporation of Special Nodes into DSCF

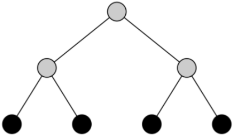
- Simplification of the metric made DSCF a potential algorithm towards a practical implementation.

- Special bit-patterns (nodes) of interest: Repetition (Rep), and Rate-1 nodes.

- Examples:



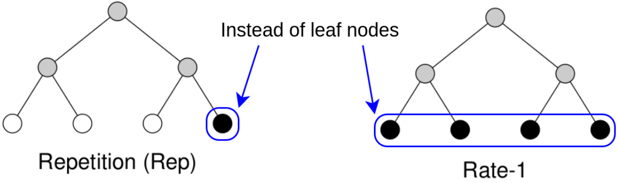
Repetition (Rep)



Rate-1

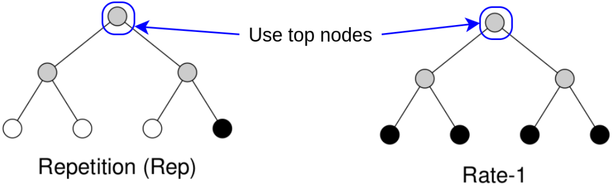
# Incorporation of Special Nodes into DSCF

- Simplification of the metric made DSCF a potential algorithm towards a practical implementation.
- Special bit-patterns (nodes) of interest: Repetition (Rep), and Rate-1 nodes.
- Examples:



# Incorporation of Special Nodes into DSCF

- Simplification of the metric made DSCF a potential algorithm towards a practical implementation.
- Special bit-patterns (nodes) of interest: Repetition (Rep), and Rate-1 nodes.
- Examples:



# Incorporation of Special Nodes into DSCF

Recall the original DSCF metric:

$$M_\alpha(\mathcal{E}_\omega) = \sum_{j \in \mathcal{E}_\omega} |L^0[\mathcal{E}_{\omega-1}]_j| + S_\alpha(\mathcal{E}_\omega)$$

# Incorporation of Special Nodes into DSCF

Recall the original DSCF metric:

$$M_\alpha(\mathcal{E}_\omega) = \sum_{j \in \mathcal{E}_\omega} |L^0[\mathcal{E}_{\omega-1}]_j| + S_\alpha(\mathcal{E}_\omega)$$

Let us split it as follows:

$$M_\alpha(\mathcal{E}_\omega) = \underbrace{|L^0[\mathcal{E}_{\omega-1}]_{i_\omega}|}_{M'(L)} + \underbrace{\sum_{j \in \mathcal{E}_{\omega-1}} |L^0[\mathcal{E}_{\omega-1}]_j| + S_\alpha(\mathcal{E}_\omega)}_{M''(L)}.$$

- $M'(L)$ is the *instantaneous value* obtained from the node, on the spot.
- $M''(L)$ is the *accummulative value*, formed over the course of the decoding.

# Incorporation of Special Nodes into DSCF

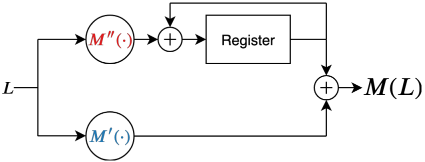
$$M(L) = M'(L) + M''(L).$$

- $M'(L)$ is directly obtained from the LLR magnitude of the index.
- $M''(L)$ is updated by the LLR magnitude of the index.
-

# Incorporation of Special Nodes into DSCF

$$M(L) = M'(L) + M''(L).$$

- $M'(L)$ is directly obtained from the LLR magnitude of the index.
- $M''(L)$ is updated by the LLR magnitude of the index.

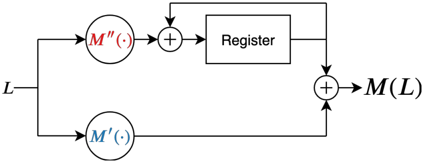



-

# Incorporation of Special Nodes into DSCF

$$M(L) = M'(L) + M''(L).$$

- $M'(L)$ is directly obtained from the LLR magnitude of the index.
- $M''(L)$ is updated by the LLR magnitude of the index.




- **Main idea:** Do the same with special nodes involved.

# Involving Rep Nodes into DSCF

- The only information bit in a Rep node is repeated over at the top-level.
- The same LLR can be obtained by summing all top-node LLRs:

$$L_{\mathrm{Rep}} = \left| \sum_{i \in N_v} L^S[\mathcal{E}_{\omega-1}]_i \right|$$

Therefore;

- $L_{\mathrm{Rep}}$ is directly used to create $M'(L)$ and update $M''(L)$.

# Involving Rate-1 Nodes into DSCF

- Rate-1 nodes are uncoded sequences (no redundancy).
- As a result, all top-node indices are considered for bit-flipping individually.

For each top-node index $i$:

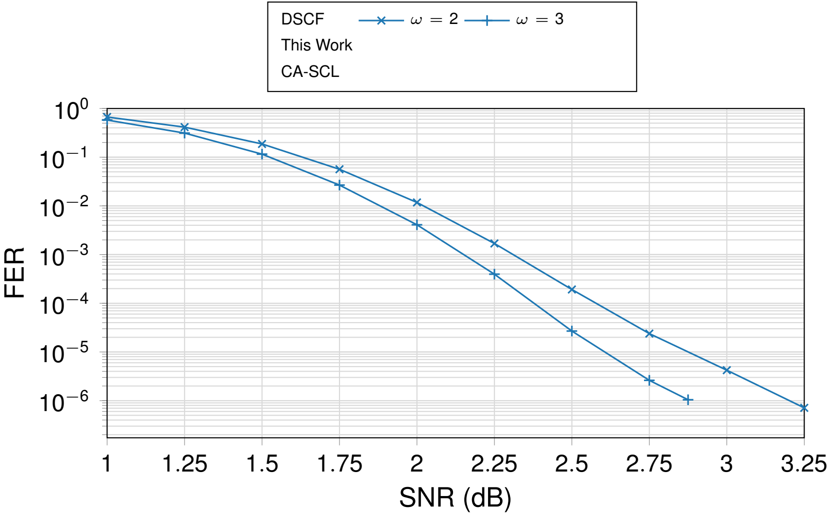
$$L_{\text{Rate-1},i} = |L^S[\mathcal{E}_{\omega-1}]_i|$$

For a Rate-1 node of size $N_v$;

- $N_v$ $M'(L)$ values are created for each top-node index.
- $M''(L)$ is updated once per Rate-1 node using $N_v$ LLRs.

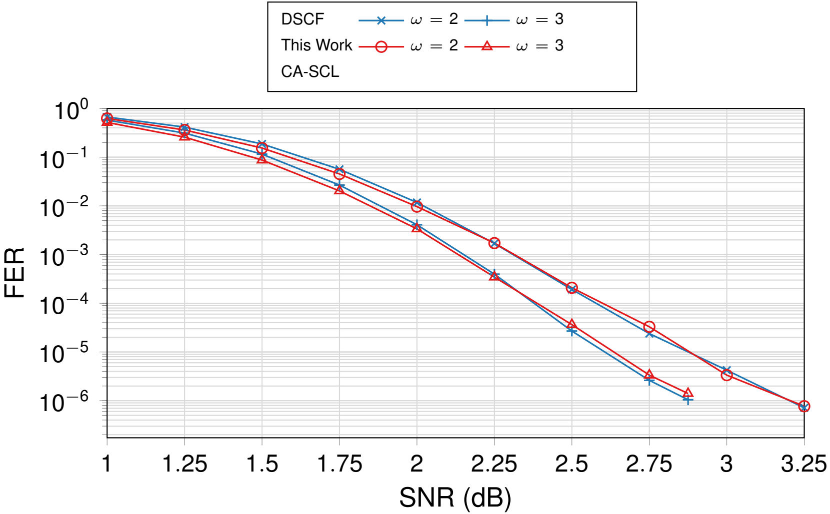
# Results - Performance

- $PC(1024, 512)$, CRC length $C = 16$ bits, $T_{\max} \in \{40, 200\}$ for $\omega \in \{2, 3\}$.



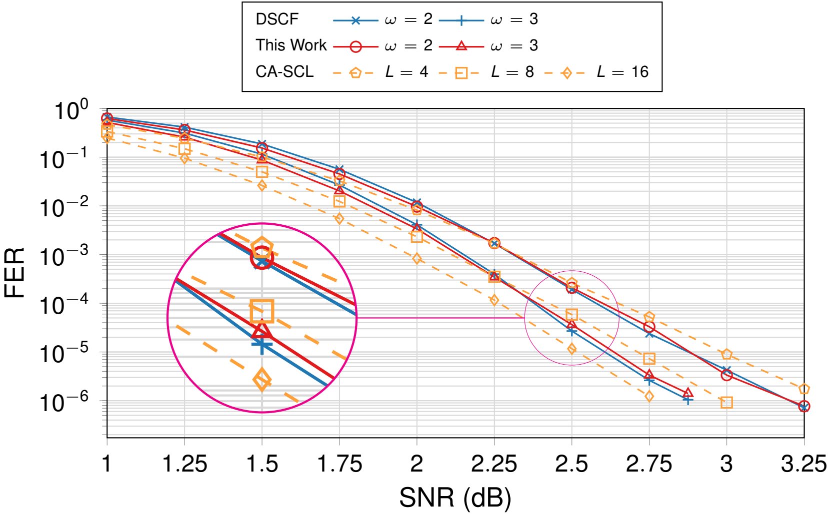
# Results - Performance

- $PC(1024, 512)$, CRC length $C = 16$ bits, $T_{\max} \in \{40, 200\}$ for $\omega \in \{2, 3\}$.



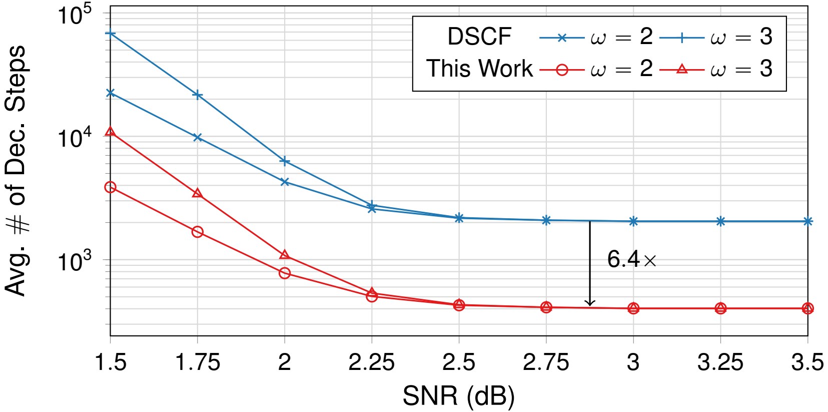
# Results - Performance

- $PC(1024, 512)$, CRC length $C = 16$ bits, $T_{\max} \in \{40, 200\}$ for $\omega \in \{2, 3\}$.



# Results - Computational Complexity

- $PC(1024, 512)$, CRC length $C = 16$ bits, $T_{\max} \in \{40, 200\}$ for $\omega \in \{2, 3\}$.



# Conclusion

- First steps towards a practical implementation for the Dynamic SCF algorithm.

# Conclusion

- First steps towards a practical implementation for the Dynamic SCF algorithm.
- We showed how to:
  - Replace transcendental computations with a simple threshold
  - Implement some special bit patterns into DSCF to speed up the decoding.

# Conclusion

- First steps towards a practical implementation for the Dynamic SCF algorithm.
- We showed how to:
  - Replace transcendental computations with a simple threshold
  - Implement some special bit patterns into DSCF to speed up the decoding.
- When all simplifications are applied:
  - The FER is equivalent to the original DSCF algorithm,
  - Expensive operations (log, exp, $\times$) are eliminated,
  - Average number of decoding steps is reduced by up to $6.4\times$.

Thank you for your attention!