基於稀疏圖與動態排程之極化碼解碼器設計
Design of a Polar Code Decoder Based on Sparse Graphs and Dynamic Scheduling

組員姓名：陳聖諦　　指導教授：翁詠祿 教授

Introduction

Polar codes are adopted as the primary channel coding scheme for 5G communication systems. However, traditional decoding methods, such as Successive Cancellation (SC) and Successive Cancellation List (SCL), suffer from high decoding latency and suboptimal bit error rate (BER) performance [2].
Transforming polar codes into LDPC-like structures, decoded with Belief Propagation (BP), can achieve lower complexity and better parallelism.
This research aims to enhance decoding efficiency and reliability by optimizing the factor graph structure and introducing improved dynamic scheduling strategies.

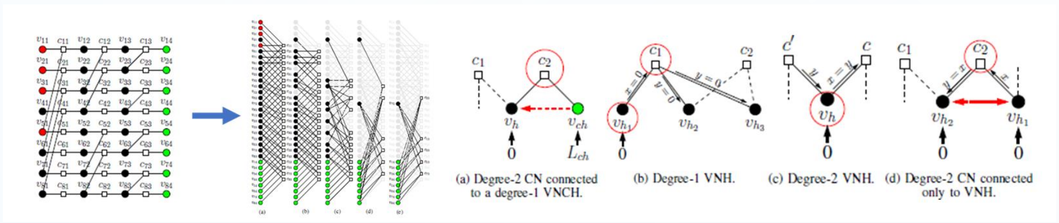
Keywords: Polar codes, Sparse graph, Residual belief propagation (RBP), Pruning algorithm, Low-density parity-check like code (LDPC-like code)

Fundamentals

Sparse Graph Pruning [2]

We perform sparse graph pruning based on node degree properties.
Frozen variable nodes (VN) and redundant check nodes (CN) are systematically removed according to specific rules:

- Frozen VN removal,
- Degree-1 CN removal,
- Condensation of VN-CH and VN-Hidden; as the figure shows.



Residual Belief-Propagation (RBP) [4]

Residual Belief Propagation (RBP) enhances convergence by dynamic scheduling message updates based on residual

$$\mathcal{R}_k(\mathbf{m}) = |f_k(\mathbf{m}) - m_k|$$

This approach focuses computational effort on parts of the graph that are less converged, achieving faster early-stage convergence and improve resistance to trapping sets.

Algorithm 2: RBP Algorithm
Input:
$\mathcal{L}(y_i) \Rightarrow$ LLR of received message $\mathcal{Y}$
Output:
$Q_{j\to i} \Rightarrow$ LLR of decoded message $\hat{\mathcal{Y}}$

1. Initialize all $r_{i\to j} = 0$
2. Initialize all $q_{j\to i} = \mathcal{L}(y_i)$
3. Compute all $\mathcal{R}_k(r_{i\to j})$ and set queue **Q**
4. **while** Stopping criteria is not satisfied
5. Find the first message $r_{i\to j}$ in **Q**
6. Generate and propagate $r_{i\to j}$
7. Set $\mathcal{R}_k(r_{i\to j}) = 0$ and re-order **Q**
8. **for** every $c_a \in \mathcal{N}(v_j)\backslash c_i$
9. Generate and propagate $q_{j\to a}$
10. **for** every $v_b \in \mathcal{N}(c_a)\backslash v_j$
11. Compute $\mathcal{R}_k(r_{a\to b})$ and re-order **Q**
12. **end for**
13. **end for**
14. **end while**

Proposed Methods

Novel Pruning Methods on Sparse Graph

To ensure graph optimization, two pruning methods were proposed:

- **Iterative-over Method**: Keep pruning until the size of **H** does not change for S rows.
- **Greedy Priority-Queue**: Given a priority-queue **Q** of pruning actions: if an action does not change the size of **H**, it suffers penalty. Keep pruning until there is no action can change the size of **H**.

An Improved RBP Algorithm

While traditional RBP prioritizes updating the message with the largest residual, it suffers from severe computational overhead due to frequent recalculations across the entire graph.
To address this, we propose:

- **Batch-RBP (or B-RBP)**

A scheduling method based on RBP.
It updates the top edge first, and if newly computed residuals exceed the top edge's, they are updated immediately. This forms a dynamic batch updates of high-residual edges during each iteration.

- **Top-K update strategy**

Like original B-RBP, in each iteration, the K highest-residual edges are selected and updated, and if newly computed residuals exceeding the K-th top edge's, they are updated immediately.

This makes the update process more focused on under-converged regions while avoiding over-updating low-priority edges.

Algorithm 2: B-RBP Algorithm
Input:
$\mathcal{L}(y_i) \Rightarrow$ LLR of received message $\mathcal{Y}$
Output:
$Q_{j\to i} \Rightarrow$ LLR of decoded message $\hat{\mathcal{Y}}$

1. Initialize all $r_{i\to j} = 0$
2. Initialize all $q_{j\to i} = \mathcal{L}(y_i)$
3. Compute all $\mathcal{R}_k(r_{i\to j})$ and set queue **Q**
4. **while** Stopping criteria is not satisfied
5. Find the first message $r_{i\to j}$ in **Q**
6. Generate and propagate $r_{i\to j}$
7. Set $\mathcal{R}_k(r_{i\to j}) = 0$ and re-order **Q**
8. **for** every $c_a \in \mathcal{N}(v_j)\backslash c_i$
9. Generate and propagate $q_{j\to a}$
10. **for** every $v_b \in \mathcal{N}(c_a)\backslash v_j$
11. Compute $\mathcal{R}_k(r_{a\to b})$ and re-order **Q**
12. **if** $\mathcal{R}_k(r_{a\to b}) > \mathcal{R}_k(r_{i\to j})$
13. propagate $r_{a\to b}$ directly
14. **end if**
15. **end for**
16. **end for**
17. **end while**

Simulation Results

Table 1. Average Complexity between different algorithm

Flooding BP	$\Theta(m)$
RBP	$\Theta(m) * \Theta(m^2)$
Batch-RBP	$\Theta(m/\gamma) * \Theta(m\log(m)), \quad \gamma \gg 1$

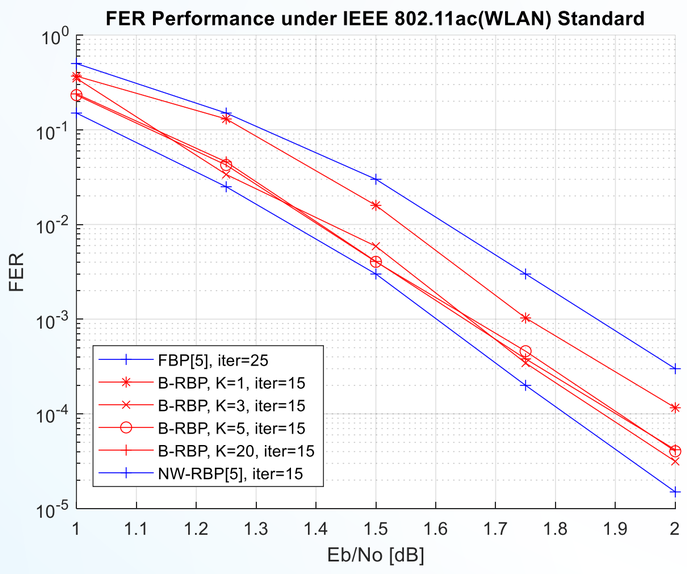


Fig 1. FER results for LDPC code under IEEE 802.11ac (WLAN) standard with batch-RBP

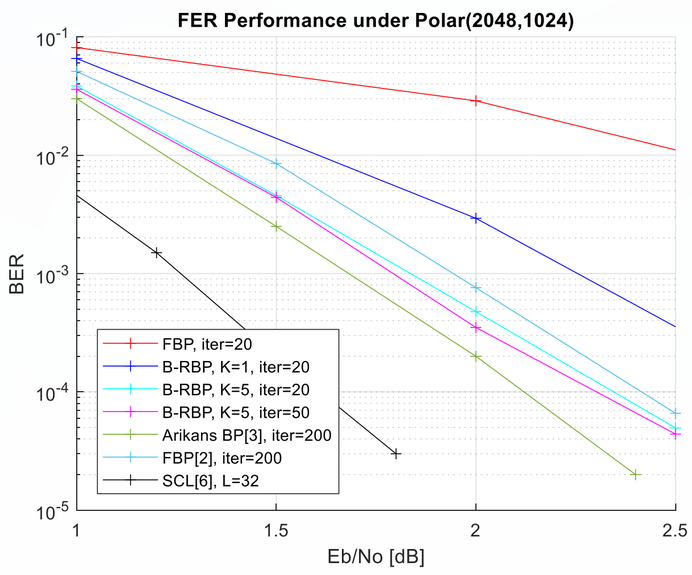


Fig 3. BER results for Polar(2048,1024) for different decoding algorithms

Impact of Graph Pruning

Graph pruning has little effect on FBP performance, indicating FBP's insensitivity to local factor graph changes. However, for B-RBP decoding, pruning significantly impacts performance. Iterative over-pruning particularly enhances local convergence and improves the overall bit error rate (BER) compared to standard pruning methods.

Performance of Batch-RBP and Top-K

Compared to conventional FBP with the same maximum iteration limit (20), B-RBP achieves about **one order of magnitude** improvement in BER at moderate to high SNR. Even against FBP with heavy iterations (up to 200), Batch-RBP maintains a performance advantage, showing the effectiveness of prioritized updates and graph optimization.

Comparison with Arikan's BP

Despite improvements, a noticeable gap remains compared to Arikan's original BP. Residual short cycles and trapping sets in the pruned graph limit convergence and error correction capabilities, whereas Arikan's BP benefits from the highly structured original polar graph.

Conclusion

This study demonstrates that optimizing the factor graph structure and introducing efficient dynamic scheduling significantly improves the decoding performance of LDPC-like polar code decoders.
The proposed methods provide a practical and effective approach for designing polar code receivers under resource-constrained scenarios.
Further optimization in graph transformation and dynamic scheduling could help narrow the performance gap with native polar code decoders in the future.

Reference

[1] E. Arıkan, "Channel Polarization: A Method for Constructing Capacity-Achieving Codes for Symmetric Binary-Input Memoryless Channels," IEEE Trans. Inf. Theory, vol. 55, no. 7, pp. 3051–3073, July 2009.
[2] Sebastian Cammerer, et al. "Sparse Graphs for Belief Propagation Decoding of Polar Codes" IEEE, International Symposium on Information Theory (ISIT). 2018.
[3] E. Arıkan, "Polar codes: A pipelined implementation," Proc. 4th ISBC, pp. 11–14, 2010.
[4] A. I. V. Casado, M. Griot and R. D. Wesel, "Informed Dynamic Scheduling for Belief-Propagation Decoding of LDPC Codes," 2007 IEEE International Conference on Communications, Glasgow, UK, 2007, pp. 932-937.
[5] H. -C. Lee and Y. -L. Ueng, "Informed dynamic schedules for LDPC decoding using belief propagation," 2013 IEEE 24th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC).
[6] S. Cammerer, "LDPC-like Decoding of Polar Codes", GitHub Repository: https://github.com/SebastianCa/LDPC-like-Decoding-of-Polar-Codes.