國立清華大學 電機工程學系

實作專題研究成果摘要

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

專題領域：通訊領域

組 別：A507

指導教授：翁詠祿 教授

組員姓名：陳聖諦

研究期間：2024年9月 至 2025年4月止，共8個月

摘要

本研究針對極化碼(Polar Codes)解碼器，提出基於稀疏圖剪枝與動態排程優化之設計方法。首先，透過分析極化碼結構，採用剪枝技術簡化因數圖(factor graph)，降低訊息傳遞中的短環(short cycles)密度。

其次，針對殘差置信度傳播(Residual Belief Propagation, RBP)演算法中存在的高計算複雜度問題，本研究提出批次式殘差置信度傳播算法(Batch-RBP)與 Top-K 更新策略，以減少重複計算負擔，同時維持優先更新未收斂訊息之特性。

實驗以 IEEE 802.11ac 標準與系統極化碼 (2048, 1024)為測試平台進行驗證。結果顯示，結合剪枝與 Batch-RBP 設計能有效提升基於 LDPC-like 極化碼解碼器之位元錯誤率(Bit Error Rate, BER)性能，特別在中高 SNR 區間約提升一個數量級；但與 Arikan's BP 及 SCL 等原生極化碼解碼器相比，仍有性能落差，推測原因為殘留短環與局部結構限制所致。本研究成果為資源受限應用中極化碼解碼器之效能優化提供具體可行之方法與參考依據。

關鍵字：極化碼、稀疏圖、殘差置信度傳播、剪枝演算法、低密度奇偶檢查碼。

章節目錄

一、 研究背景

第五代行動通訊技術(5G)採用極化碼(Polar Codes)作為主要編碼技術。極化碼透過特殊編碼結構，將通道極化成「純噪聲」與「無噪聲」兩類，並隨編碼長度增加而趨於完全極化。利用近似無噪聲的通道進行資料傳輸，可實現高吞吐量與低錯誤率[1]。

現有極化碼解碼技術面臨兩大挑戰：(1) 基於順序消除(SC)的演算法具有較差的位元錯誤率(BER)表現；(2) SC 與順序消除列表(SCL)演算法存在延遲過大的問題。相較之下，將極化碼轉換為類低密度奇偶檢查碼(LDPC-like code)，並以置信度傳播(Belief Propagation, BP)演算法解碼，可兼顧較佳錯誤率與較低硬體複雜度[2]。此方法透過校驗矩陣轉換與剪枝，降低短環密度，達到接近 Arikan's BP [3]的性能表現。

然而，文獻[2]中以和積演算法(Sum-Product Algorithm, SPA)搭配 flooding 排程進行 BP 解碼，存在收斂速度慢與陷阱集處理不佳的問題。由於剪枝後校驗矩陣仍殘留短環，整體解碼性能仍受限制。針對此情形，文獻[4]提出殘差置信度傳播(Residual Belief Propagation, RBP)，依據訊息殘差大小動態調整更新順序，加速收斂並改善陷阱集錯誤。

二、 研究目的

本研究旨在延續上述方法，優化 LDPC-like 解碼器在算法層面的性能。分為兩個階段：

第一階段旨在重現[2]的算法與驗證，包含：編碼/解碼平台的搭建、剪枝算法實現與結果驗證；另外，也探討不同剪枝的貪婪策略對糾錯性能造成的影響。

第二階段，分別使用 SPA 與 RBP 算法解碼此 LDPC-like 碼，並在 RBP 架構上提出一種改良算法——批量 RBP (Batch-RBP)。相比於傳統的 RBP 算法。此演算法透過合併更新與邊選擇機制，在不顯著影響錯誤率的情況下大幅降低運算複雜度；運用在 LDPC-like 的解碼器中，相比於 SPA 算法，錯誤率性能得到顯著的改善。

三、 研究方法

3.1 編碼/解碼平台的搭建

置信度傳播解碼器(Belief Propagation, BP)為目前常用之解碼演算法之一。對於(N, K)-LDPC 碼，可透過坦納圖(Tanner Graph)表示其奇偶檢查矩陣中檢查節點(Check Nodes, CNs)與變數節點(Variable Nodes, VNs)之連結。訊息在 BP 演算法中於 CN 與 VN 之間交互傳遞，持續疊代直至解碼完成。

LDPC 之編碼：對於一個碼長為 K 的二進位訊息$\mathcal{U} = (u_1, u_2, u_3 \dots u_K),\ u_j \in GF(2), \forall j$，

經由生成矩陣 **G** 進行編碼$\mathcal{X} = \mathcal{U}\mathbf{G}$，可以得到碼長為 N 的二進位原始碼字$\mathcal{X} = (x_1, x_2, x_3 \dots x_N),\ x_i \in GF(2), \forall i$，進行二位元相位調變(Binary Phase-Shift Keying, BPSK)後可以得到碼字$\overline{\mathcal{X}} = \{\overline{x_i} | \overline{x_i} = \begin{cases} -1, if\ x_i = 1 \\ 1, if\ x_i = 0 \end{cases}, \forall i\}$ 並從 TX 端發出。在經過可加性高斯白雜訊通道(additive white Gaussian noise, AWGN)後於 RX 端收到碼字$\mathcal{Y} = \overline{\mathcal{X}} + N(0, \sigma^2)$，以模擬自然的噪聲干擾。

LDPC 之解碼：當解碼器收到通道傳來的碼字$\mathcal{Y}$，會先將其轉換成對數似然率(log-likelihood ratio)形式後，將送入對應的 VN

$$\mathcal{L}(y_i) = \log\left(\frac{P(c_i = 0|\mathcal{Y})}{P(c_i = 1|\mathcal{Y})}\right) = \frac{2y_i}{\sigma^2}$$

**我們以$r_{i\to j}$表示從c_i傳遞至v_j的訊息、$q_{j\to i}$表示從v_j傳遞至c_i的訊息。接下來 VN 根據當前紀錄的訊息傳遞至 CN，進行以下計算傳遞至$\mathcal{N}(c_i)$：

$$r_{i\to j} = 2tanh^{-1}\left(\prod_{v_j' \in \mathcal{N}(c_i)\setminus v_j} \tanh\left(\frac{1}{2} q_{j'\to i}\right)\right) \tag{1}$$

VN 收到後進行以下更新，傳遞回 CN，反覆疊代運算：

$$q_{j\to i} = \mathcal{L}(y_i) + \sum_{c_i' \in \mathcal{N}(v_j)\setminus c_i} r_{i'\to j} \tag{2}$$

此時糾錯碼字的後驗對數似然比為：

$$Q_{j\to i} = \mathcal{L}(y_i) + \sum_{c_i \in \mathcal{N}(v_j)\setminus c_i} r_{i\to j} \tag{3}$$

透過硬判決，可以得到碼字$\hat{y_i}$

$$\hat{y_i} = \begin{cases} 1, & if\ Q_{j\to i} < 0 \\ 0, & otherwise \end{cases} \tag{4}$$

此時經由校驗矩陣 H 可以檢查：$H\hat{\mathcal{Y}} \triangleq 0$，若為0，則解碼成功；若不為0，則解碼失敗，並從公式(1)開始重新疊代；直到疊代次數耗盡。

依據上述流程，搭建完成編碼/解碼測試平台(如圖3-1所示)，能以模組化方式替換子模塊，便於不同演算法之性能比較。

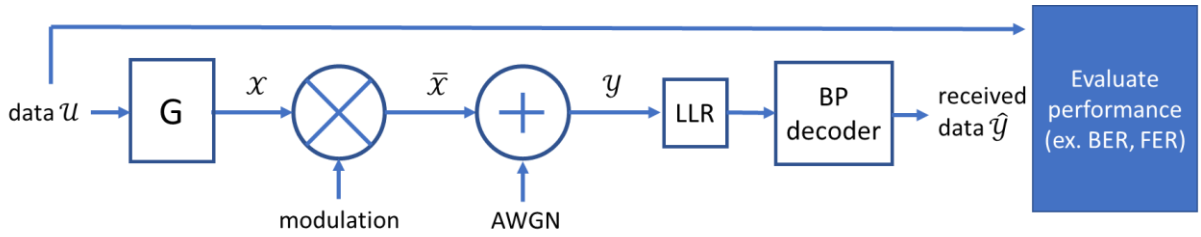

圖 3-1 編碼/解碼測試平台架構圖

3.2 剪枝算法實現

為降低極化碼於 Belief Propagation 解碼中的運算與記憶體負擔，本研究依據文獻[2]方法，實作針對極化碼原生 factor graph 之剪枝演算法。透過層層展開並插入 Variable Nodes (VNs)與 Check Nodes (CNs)，可將原極化碼結構轉換為單層的稀疏 Tanner graph，產生適用於 LDPC-like 解碼的稀疏校驗矩陣(如圖3-2)。

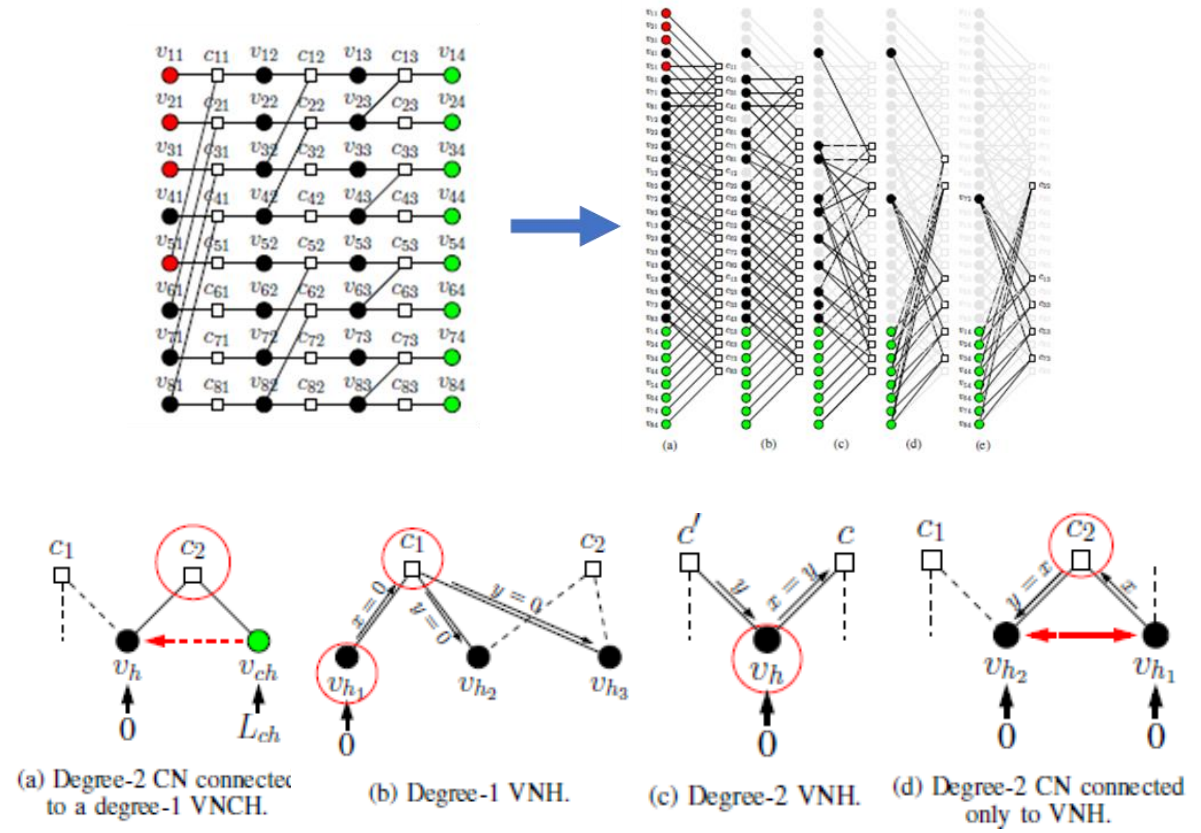

圖 3-2 稀疏矩陣的產生與 Factor Graph 之無效連接情況

在此 factor graph 中，節點可分為三類：通道 VN(VNCH，綠色)、隱藏 VN(VNH，黑色)、及凍結 VN(Frozen Node，紅色)。根據節點連接情形，可定義以下無效結構並加以移除或合併：

(1) Frozen node：其 log-likelihood ratio (LLR) 固定為無限大，訊息不會因更新而變動，

故可直接從圖中移除。

(2) Degree-1 CN：即某個 CN 僅連接到一個 VN，從此 CN 傳遞出的訊息只在 VN=0存在約束並且只能是無限大，因此亦可連同其相連 VN 一併移除。

(3) Degree-1 VNCH and Degree-2 CN：某個 VNCH 僅連接到一個 CN，再連接到一個 VNH。這個 CN 就是一個多餘的傳遞節點，無法帶來任何訊息；故可以移除此 CN，並將 VNH 替換為 VNCH (見圖3-2-(a))。

(4) Degree-1 VNH：即某個 VNH 僅有連接到一個 CN。由於 VNH 能帶來的額外訊息為 0，故可直接移除此 VNH (見圖3-2-(b))。

(5) Degree-2 VNH：即某個 VNH 僅有連接到2個 CN。類似於情況(3)，訊息從 c'傳遞到 c，其中 vh 無法帶來任何訊息；故可以移除 vh，並將 c'與 c 合併(見圖3-2-(c))。

(6) Degree-2 CN：即某個 CN 僅有連接到2個 VNH。與情況(5)相同，這無法帶來任何額外訊息；故可以移除此 CN，合併2個 VNH(見圖3-2-(d))。

透過貪婪算法，持續剪枝直到矩陣大小不再變化，剪枝結束[註1][7]。其算法如下：

Algorithm 1: Prune **H**-matrix

Input:

$\mathbf{H}_{orig}$. ⇒ original **H**-matrix

Output:

$\mathbf{H}_{pruned}$ ⇒ pruned **H**-matrix

1. $\mathbf{H}_{pruned} \leftarrow \mathbf{H}_{orig}$
2. Remove Frozen VN from $\mathbf{H}_{pruned}$
3. Apply all pruning operations on $\mathbf{H}_{pruned}$.
4. **if** size($\mathbf{H}_{pruned}$) not change **then**
5. return $\mathbf{H}_{pruned}$
6. **else then**
7. go to 3.
8. **end if**

3.3 RBP 算法實現

傳統 BP 解碼多採用同步(flooding)或固定排程(Standard Sequential Scheduling, SSS)，但收斂速度受限於更新順序。為提升效率，文獻[4]提出殘差置信度傳播(Residual Belief Propagation, RBP)，屬於動態排程(Informed Dynamic Scheduling, IDS)策略。

RBP 核心理念為：於每步選取殘差(Residual)最大的訊息更新，其中殘差定義為當前訊息與更新後訊息間的差異。假設某一條訊息m_k對應的訊息更新函數為$f_k(\mathbf{m})$，則其殘差定義為：

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

LDPC 解碼中的訊息更新遵循檢查節點(CN)與變數節點(VN)之間的標準規則，並以殘差量化更新緊迫性。傳統 RBP 流程如 Algorithm 4所示，優先處理收斂緩慢區域以加速整體疊代。

Algorithm 2: RBP decoding for LDPC code

Input:

$\mathcal{L}(y_i)$ ⇒ LLR of received message $\mathcal{Y}$

Output:

$Q_{j\to i}$ ⇒ 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 generate queue **Q**
4. **while** Stopping criteria is not satisfied **then**
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**

四、 研究結果

4.1 批次 RBP

雖然傳統 RBP 能夠有效加速收斂，但在實作時仍存在明顯的計算浪費。為了解決此問題，本研究提出**批次 RBP（Batch-RBP, 或簡稱 B-RBP）**，以降低重複計算所帶來的複雜度，同時保留原本 RBP「優先處理未收斂區域」的貪婪策略。

在 Batch-RBP 中，每一次主迴圈(while loop)中，不再僅更新一條邊，而是允許更新多條訊息邊，以提高運算效率。具體做法如下：

1. 首先，找到當前 **Q** 中剩餘量最大的邊 E_1，其殘差為$\mathcal{R}_1$。
2. 立刻更新此邊 E_1。
3. 在隨後的殘差重新計算過程中，若出現任一邊 E_2 計算出的新殘差$\mathcal{R}_2 > \mathcal{R}_1$，則立即更新$E_2$，不等待下輪。
4. 持續進行，直到整張圖的殘差計算完成。

Algorithm 2-a: Batch-RBP decoding for LDPC code

Input:

$\mathcal{L}(y_i)$ ⇒ LLR of received message $\mathcal{Y}$

Output:

$Q_{j\to i}$ ⇒ 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 generate queue **Q**
4. **while** Stopping criteria is not satisfied **then**
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** $\mathcal{R}_k(r_{a\to b}) > \mathcal{R}_k(r_{i\to j})$ **then**
13. propagate $r_{a\to b}$ directly
14. **end if**
15. **end for**
16. **end for**
17. **end while**

可以觀察到，當所有的新殘差皆大於$\mathcal{R}_1$，這個算法將退化至 flooding；而當所有新殘差皆小於$\mathcal{R}_1$，這個算法將退化至 RBP。我們可以計算各算法的平均複雜度，假定 m 為校驗矩陣中之非零元素數量，結果如表4-1：

表 4-1 不同解碼算法之平均複雜度比較

解碼算法	平均複雜度(單次疊代)
Flooding BP	$\sim \Theta(m)$
RBP	$\sim \Theta(m) * \Theta(m^2)$
Batch-RBP	$\sim \Theta(m/\gamma) * \Theta(mlog(m)), \quad \gamma \gg 1$

可以看到此算法相比於傳統 RBP 複雜度顯著降低。此策略保持了 RBP 中「針對尚未收斂區域優先反應」的特性，同時大幅減少單次只更新一條邊所帶來的重複計算負擔。

為了進一步細緻控制每一輪的更新行為，本研究在 Batch-RBP 基礎上，提出 **Top-K 批次更新策略**。其思路如下：

1. 首先，找到當前 **Q** 中最大的前 K 條邊（即$E_1, E_2 \dots E_K$），其共同特徵是殘差大於等於某一門檻$\mathcal{R}_K$。
2. 立刻更新這 K 條邊。
3. 接下來在新的訊息計算與更新過程中，只要出現任何新的邊E_n具有殘差$\mathcal{R}_n > \mathcal{R}_K$，則立即更新$E_n$，不等待下輪。
4. 持續進行，直到整張圖的殘差計算完成。

Algorithm 2-b: Batch-RBP with Top-K strategy

Input:

$\mathcal{L}(y_i)$ ⇒ LLR of received message $\mathcal{Y}$

K ⇒ # of edge picked per loop

Output:

$Q_{j\to i}$ ⇒ 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 generate queue **Q**
4. **while** Stopping criteria is not satisfied **then**
5. Find the first **K** messages $\{r_{i_1\to j_1}, \dots r_{i_K\to j_K}\}$ in **Q**
6. Generate and propagate $\{r_{i_1\to j_1}, \dots r_{i_K\to j_K}\}$
7. Set all $\mathcal{R}_k(r_{i\to j}) = 0$ and re-order **Q**
   ** *say* $\mathcal{N}(v_j) = \mathcal{N}(v_{j_1}) \cap \dots \cap \mathcal{N}(v_{j_K})$
8. **for** every $c_a \in \mathcal{N}(v_j) \backslash \{c_{i_1}, \dots c_{i_K}\}$
9. Generate and propagate $q_{j\to a}$
10. **for** every $v_b \in \mathcal{N}(c_a) \backslash \{v_{j_1}, \dots v_{j_K}\}$

11. Compute $\mathcal{R}_k(r_{a\to b})$ and re-order **Q**
12. **if** $\mathcal{R}_k(r_{a\to b}) > \mathcal{R}_k(r_{i\to j})$ **then**
13. propagate $r_{a\to b}$ directly
14. **end if**
15. **end for**
16. **end for**
17. **end while**

Top-K 策略相較於基本 B-RBP 更加「謹慎地」選擇初始更新組，儘管運作速度稍慢，但整體收斂品質有所提升，特別在低 SNR 的情境中，表現出更好的解碼效果。

為評估批次 RBP(Batch-RBP)演算法性能，本研究基於文獻[4]提出之 RBP 算法，並參考文獻[5]之模擬條件進行驗證。採用 IEEE 802.11ac 標準 LDPC 碼(碼長1944、碼率 1/2)進行模擬，並以不同 Top-K 設定之 Batch-RBP 與 Flooding BP(FBP)、Node-wise RBP(NW-RBP)進行比較[註2]。

實驗結果如圖4-1所示，相同疊代次數下，B-RBP 相比於 FBP 能有效降低區塊誤碼率 (Frame Error Rate, FER)，顯示批次式貪婪更新有助於加速收斂。惟因每輪同時更新多個訊息，B-RBP 在最終性能上略遜於更新更謹慎的 NW-RBP。

隨 Top-K 值增加，B-RBP 性能提升，但提升幅度逐漸趨緩，且過高的 K 值反而可能因更新過多低殘差訊息而影響效率。綜合觀察，最佳 Top-K 值約落於3至5，可兼顧計算複雜度與性能表現。整體而言，B-RBP 在保有原 RBP 加速收斂特性的同時，降低了單次更新的運算負擔，適合於硬體資源受限之應用。

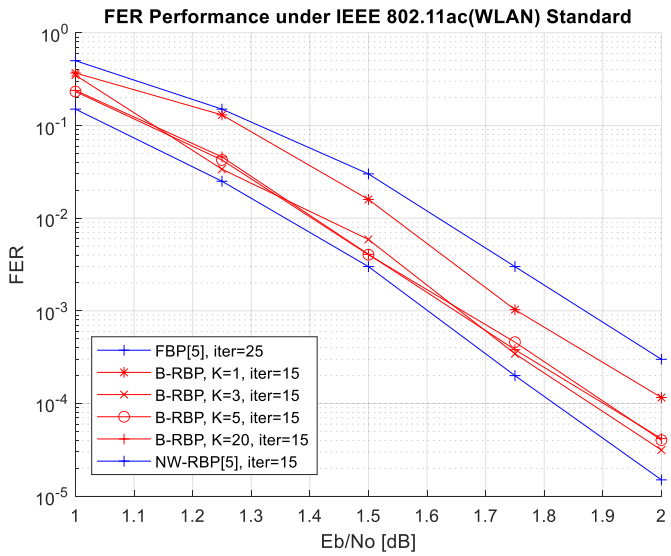

圖 4-1 B-RBP 與 NW-RBP 之性能比較

[註2]：由於文獻中缺乏同時提供 FBP 與 RBP 性能數據資料，故本研究參考含 FBP 與 NW-RBP 之文獻進行比較。根據[4]，NW-RBP 與標準 RBP 性能差異甚微，可合理作為近似基準。

4.2 剪枝算法之優化

本研究基於文獻[2]的剪枝算法設計兩種剪枝策略。

1. **無條件多次剪枝(iterative over-pruning)**：即使矩陣大小暫時不變，仍持續剪枝，僅當連續 **S** 次剪枝後皆無變化時才終止，避免因局部停滯提早中止。算法如下：

Algorithm 1-a: Prune **H**-matrix by iterative over-pruning

Input:

$\mathbf{H}_{orig}$ ⇒ original **H**-matrix

S ⇒ max # of successive pruning

Output:

$\mathbf{H}_{pruned}$ ⇒ pruned **H**-matrix

1. $\mathbf{H}_{pruned} \leftarrow \mathbf{H}_{orig}$
2. Remove Frozen VN from $\mathbf{H}_{pruned}$
3. Apply all pruning operations on $\mathbf{H}_{pruned}$
4. **if** size($\mathbf{H}_{pruned}$) not change in **S** row **then**
5. return $\mathbf{H}_{pruned}$
6. **else** go to 3.
7. **end if**

---

2. **優先佇列剪枝(Greedy Priority-Queue pruning)**：將所有剪枝動作(action)放入優先佇列中，依據每次執行後矩陣縮減程度更新排名，優先選取最有效的剪枝操作；若所有動作皆無法改變矩陣大小，則結束剪枝。算法如下：

Algorithm 1-b: Prune **H**-matrix by priority queue

Input:

$\mathbf{H}_{orig}$. ⇒ original **H**-matrix

Output:

$\mathbf{H}_{pruned}$ ⇒ pruned **H**-matrix

1. $\mathbf{H}_{pruned} \leftarrow \mathbf{H}_{orig}$
2. Remove Frozen VN from $\mathbf{H}_{pruned}$
3. Initialize priority queue **Q** of actions
4. Pick an action A_i with highest score (highest ranking)
5. $\mathbf{H}_{pruned} \leftarrow A_i(\mathbf{H}_{pruned})$
6. **if** size($\mathbf{H}_{pruned}$) does not change **then**
7. A_i suffers penalty, and its ranking drops.

8. **end if**
9. **if** all actions does not change size($\mathbf{H}_{pruned}$) **then**
10. **return** $\mathbf{H}_{pruned}$
11. **else** go to 4.
12. **end if**

[註1]: 此部分算法參照[7]，將作者的 MATLAB code 移植至 C++環境，並做了算法複雜度之優化，使之運算效率提升約20倍。

為探討剪枝策略對 BP 解碼性能之影響，本研究採用 systematic Polar code (2048, 1024)進行測試，於相同通道環境下，針對三種剪枝方法——優先佇列剪枝(Greedy Priority-Queue, Greedy-PQ)、無條件重複剪枝(iterative-over method)、與 S. Cammerer 原版剪枝——搭配 FBP 與 Batch-RBP 解碼器進行比較，固定疊代次數為20次。

實驗結果如圖4-2所示。對於 FBP 解碼而言，不同剪枝策略影響甚微，顯示 Flooding BP 對局部結構變動敏感度低。然而在 Batch-RBP 解碼下，剪枝策略影響明顯，其中無條件重複剪枝能顯著提升 BER 性能。
推測其原因，無條件重複剪枝能更均勻地應用各種剪枝操作，避免局部貪婪選擇導致結構僵化，並防止因暫時性矩陣穩定而提早終止剪枝，有助於因數圖持續優化。此特性對仰賴局部結構變化提升收斂效率之 Batch-RBP 演算法尤為重要。適當剪枝策略對於強烈依賴結構動態調整的解碼器具有明顯性能提升潛力。

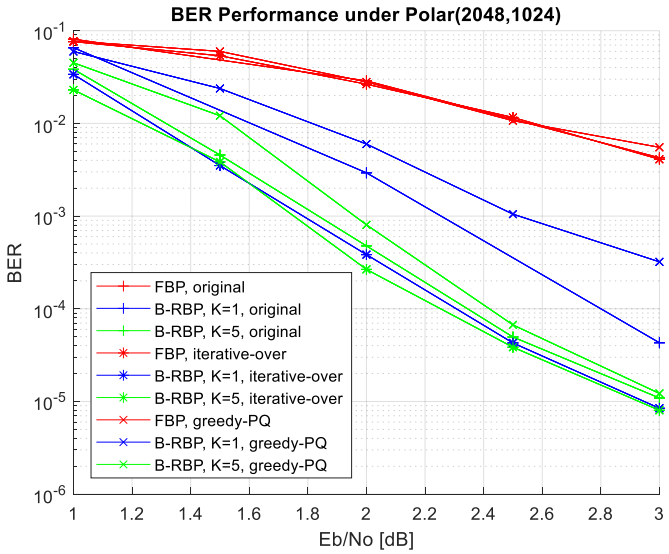

圖 4-2 不同剪枝算法之性能比較

4.3 綜合比較

為全面評估批次 RBP 與剪枝方法之整體效果，本節將模擬結果與文獻[2]中之 Flooding BP(FBP)、Arikan's BP 及 Successive Cancellation List(SCL)解碼器進行比較，使用 systematic Polar code (2048, 1024)，以位元錯誤率(BER)曲線作為指標，剪枝則採用無條

件重複剪枝產生之 H 矩陣。

如圖4-3所示，結合 B-RBP 與優化剪枝後，相較於20次疊代的 FBP 可提升約一個數量級以上之 BER 表現，即使對比疊代200次之 FBP 仍具優勢，驗證訊息更新與因數圖優化對 LDPC-like 極化碼解碼器之成效。

然而，相較於 Arikan's BP，即使提升至 50 次疊代，改進後之 B-RBP 解碼器仍存在性能差距。推測原因在於，即便經過剪枝優化，轉換過程中仍不可避免地殘留短環(short cycles)與陷阱集(trapping sets)，影響局部收斂性；而 Arikan's BP 因保有原生極化結構，於錯誤矯正上具天然優勢。綜合而言，本研究提出之改良方法在中高 SNR 區間可有效提升性能，但於極高可靠度需求下仍有進一步優化空間。

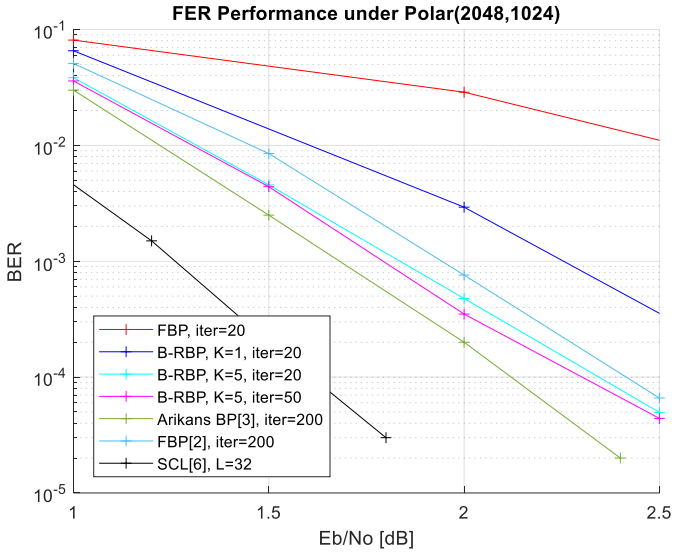

圖 4-3 綜合不同算法之性能比較

五、 總結

本研究基於稀疏圖與動態排程，設計並實現適用於極化碼之 LDPC-like 解碼器，涵蓋剪枝演算法優化、殘差置信度傳播(RBP)改良，以及新提出之批次式 RBP(Batch-RBP, B-RBP)與 Top-K 更新策略。

實驗結果顯示，剪枝能有效簡化因數圖結構，在不顯著影響 Flooding BP 性能下，明顯提升 B-RBP 解碼器表現。特別是無條件重複剪枝策略，進一步改善局部收斂性，抑制早期停滯。同時，B-RBP 與 Top-K 策略大幅降低了傳統 RBP 的重複計算負擔，維持疊代初期加速收斂的特性。綜合比較指出，整體設計相較傳統 FBP 可提升約一個數量級的錯誤率表現，但與 Arikan’s BP 或 SCL 等原生極化碼解碼器仍存性能差距，推測主因為短環與陷阱集殘留所致。

綜合而言，本研究證實了針對 LDPC-like 極化碼解碼器進行因數圖與排程優化，能有效

提升性能與降低複雜度，為資源受限應用中之極化碼接收器設計提供可行方向。未來亦可針對因數圖優化與動態排程機制進一步改良，以追求更高性能的極化碼接收器設計。

六、 參考文獻

[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] I. Tal and A. Vardy, “List Decoding of Polar Codes,” *IEEE Trans. Inf. Theory*, vol. 61, no. 5, pp. 2213–2226, May 2015.

[7] S. Cammerer, "LDPC-like Decoding of Polar Codes", GitHub Repository, Available: https://github.com/SebastianCa/LDPC-like-Decoding-of-Polar-Codes.

七、 心得感想

本專題以極化碼之 LDPC-like 解碼器設計為核心，初期由於並非通訊專業出身，對相關領域知識感到生疏，剛開始進行時常有困惑與摸索。隨著研究推進，透過實作與修正逐步掌握了編碼結構、訊息傳遞與剪枝優化等關鍵知識，對通訊系統設計建立了更具體的理解。

專題期間，與指導教授的討論較為嚴謹，也曾因準備不足被指正，但在不斷修正與嘗試中逐漸適應專業溝通的節奏，提升了問題分析與表達的精確度。

模擬驗證則是整個過程中最具趣味性的部分：每次設計改良後，投入數小時跑模擬結果，像是開寶箱般充滿期待；結果正確時帶來成就感，失敗時則促使持續調整與思考，使專題過程多了一份挑戰與樂趣。

總體而言，本專題讓我在非本專業領域中從陌生到逐步掌握，不僅增強了技術能力，也累積了在壓力中推進與修正的經驗，為未來更廣泛的學習與應用打下基礎。