A-006

検査行列の構造を利用したLDPC符号のパンクチャ法に関する一考察

A Note on a Puncturing Method for LDPC Codes Based on the Structure of Parity-Check Matrix

長田佳史* 細谷剛† 後藤正幸‡
Keishi Osada Gou Hosoya Masayuki Goto

1 はじめに

低密度パリティ検査（Low-Density Parity-Check：LDPC）符号と 確率伝播型（Belief-Propagation：BP）復号法の組み合わせ [1], [2] は，シャノン限界に迫る高い誤り訂正能力を発揮し，近年盛んに研究が行われている．

誤り訂正符号は符号化率と訂正能力がトレードオフの関係にあるため，通信路の状態により適切な符号化率をもつ符号を用いる必要があるが，符号化率ごとに符号を用意することは効率的ではない．そのため，符号語ビットをパンクチャし送信しないことで，パンクチャしたビット数に応じて符号化率を自由に設定する方法が研究されている．この際，パンクチャするビットの順番を予め決めておくことで，1 つの符号化器と復号器の組を用意するだけで符号化率を可変にすることができる．LDPC 符号に対しこのようにパンクチャして得られた符号は，符号化率が可変なパンクチャド LDPC (RCP-LDPC) 符号 [3]-[7] と呼ばれる．また，パンクチャした符号語ビットを順次追加して訂正能力を高める Incremental Redundancy Hybrid Automatic Repeat reQuest などの方式に LDPC 符号を適用する研究もなされている [10]．通常 LDPC 符号は確率的に構成されるが，extended Irregular Repeat Accumulate 符号は検査行列の一部を確定的に構成し，効率的な符号化が可能な性能を併せ持つ E^2RC 符号も提案されている [8]．

RCP-LDPC 符号を構成するにあたっては BP 復号法を仮定し，効果的なパンクチャドビットを選択しその順番を決定することで高い性能が得られる．性能の高い RCP-LDPC 符号を構成するためには，パンクチャドビットが復元されるまでの繰り返し回数 (復元レベル) を小さくするパンクチャ手法 [4] がもっとも良く知られている．一般的にパンクチャドビット数が増加すると復元レベルも増加するため，復元レベルを小さく抑えるとあまり多くのビットをパンクチャできないことがある．一方，できるだけ多くのパンクチャドビットを選択するためには，パンクチャ候補となる符号語ビットの集合が復元できるか繰り返し調べながら探索するとよい．効率よく探索するために，パンクチャを行う候補となるビットの集合の中にピボット [9] と呼ばれる復元に影響する符号語ビットを除外しながら探索する手法を応用することができる．しかし，元々パンクチャを対象とした手法ではないため，パンクチャド符号の性能に影響する復元レベルを考慮していない．

そこで著者らは，復元レベルを小さく抑えつつ，高い符号化率までパンクチャ可能な RCP-LDPC 符号に対する効果的なパンクチャドビットの選択法を提案している [5]．高い符号化率までパンクチャ可能で，かつ復号性能にも優れた符号を構成するため，提案手法では検査行列の構造を利用することで復元レベルが大きくならないように候補となるビットを選択し，全てのパンクチャドビットが復元されるまでに要する繰り返し回数の総和を評価関数として導入している．

本稿では様々な切り口から，計算機シミュレーションによる実験を行うことにより，提案手法の性能について評価を行う．その結果，提案手法によって符号化率が可変な範囲は大きく，復号性能が良好な符号が得られることを示し、様々な観点から性能について考察する．

2 準備

2.1 RCP-LDPC 符号と通信路モデル

LDPC 符号 [1], [2] は非零要素が非常に少ない M 行 N 列の疎な検査行列 $H = [H_{mn}], m = 1, 2, \ldots, M, n = 1, 2, \ldots, N$, $N > M$ により定義される符号である．本研究では簡単のため，2 元 LDPC 符号を対象とする．H に対して，符号語 $\boldsymbol{x} = (x_1, x_2, \ldots, x_N) \in F_2^N$ は，$H \cdot \boldsymbol{x}^{\mathrm{T}} = \boldsymbol{0}$ を満足する．ここで，F_2 は 2 元ガロア体上の要素 $\{0, 1\}$ を表し，ベクトル $\boldsymbol{x}^{\mathrm{T}}$ はベクトル $\boldsymbol{x}$ の転置を表す．

LDPC 符号のパリティ検査行列 H は各列を変数ノード，各行をチェックノードとした 2 部グラフ (タナーグラフ) で表すことができ，変数ノード n とチェックノード m は，$H_{mn} = 1$ のとき枝で接続される．ここで，各ノードから出る枝の本数を次数と呼び，各変数ノード，各チェックノードの次数がそれぞれ一定である符号を正則 LDPC 符号，一定でない符号を非正則 LDPC 符号と呼ぶ．非正則 LDPC 符号は次数分布多項式 $\lambda(x) = \sum_{i=c_{\min}}^{c_{\max}} \lambda_i x^{i-1}$, $\rho(x) = \sum_{i=d_{\min}}^{d_{\max}} \rho_i x^{i-1}$ によって定義される．λ_i (ρ_i) はタナーグラフの枝の総数に対する次数 i の変数ノード（チェックノード）から出る枝の総数の比率を表し，$c_{\max}$ ($c_{\min}$) は変数ノードの最大 (最小) 次数，$d_{\max}$ ($d_{\min}$) はチェックノードの最大 (最小) 次数を表す．この次数分布は非正則 LDPC 符号の性能を大きく左右し，一般的に良い次数分布は正則 LDPC 符号よりも復号性能が良くなることが知られている．

また，符号化率 R_0 は $R_0 = \frac{N-M}{N} = 1 - \frac{M}{N}$ と表される[1]．符号化率 R_0 の符号から $P(< M)$ ビットをパンクチャ [3] することで符号化率 R の符号を構成することができる．ここで，$R = \frac{N-M}{N-P}$ と表され，符号化率 R の符号を構成するために必要なパンクチャドビットの数 T は $T = \lfloor N(R - R_0)/R \rfloor$ である．ここで実数 A に対して $\lfloor A \rfloor$ は A 以下の最大の整数を表す記号である．RCP-LDPC 符号を構成するにあたって

* 早稲田大学大学院創造理工学研究科
† 早稲田大学理工学総合研究所
‡ 早稲田大学理工学術院

[1] ここで述べている符号化率は設計符号化率を指す．これ以降は全て設計符号化率を符号化率として扱う．

は，最大でパンクチャするビット数とパンクチャする位置及び順番を決めておくことで様々な符号化率に対応して符号が構成できる．

LDPC 符号に対し BP 復号法 [1], [2] を実行するときの各ビット位置 $n = 1, 2, ..., N$ における対数尤度比 (LLR) を

$$L_n = \log \frac{\Pr(y_n|x_n = 0)}{\Pr(y_n|x_n = 1)}, \tag{1}$$

と表す．符号語ビット x_n をパンクチャした場合，$L_n = 0$ であり，送信したビットが 0 か 1 か判定できないため，BP 復号法の実行過程においてパンクチャドビットのメッセージを非零に更新することで送信ビットを復元する必要がある．BP 復号法の特徴から，パンクチャドビットの位置に対応する変数ノード (パンクチャド変数ノード) v と接続するチェックノードのうち 1 つでも v 以外のパンクチャド変数ノードと接続していないとき v は復元される．

全てのパンクチャド変数ノードが復元可能である場合，以下で定義する復元レベルと Survived チェックノードによって分類できる．

定義 1 BP 復号法の繰り返し $i(\geq 0)$ 回目で復元されるパンクチャド変数ノードを i -SR ノードと呼ぶ．ただし 0-SR ノードはパンクチャされていない変数ノードを表す． □

定義 2 BP 復号法の繰り返し i 回目においてパンクチャド変数ノードを復元するチェックノードを i Survived チェックノードと呼び，全ての i Survived チェックノードを単に Survived チェックノードと呼ぶ． □

明らかに i -SR ノードは 1 つ以上の i Survived チェックノードをもつことがわかる．RCP-LDPC 符号の復号性能を高めるには，全てのパンクチャド変数ノードを少ない繰り返し回数で復元させると良いことが知られている [4], [6].

2.2 復元レベルに基づくパンクチャドビットの選択法

復号性能に優れた RCP-LDPC 符号の構成法として J.Ha らの復元レベルに基づくパンクチャドビットの選択法 [4] がある．Ha らは復号性能の優れた構成法として，パンクチャド変数ノードを選択する際に，既に選択したパンクチャド変数ノードの復元レベルが大きくならないようなパンクチャド変数ノードの選択法を提案した．この選択法は，パンクチャする位置を決定する Grouping アルゴリズムと符号化率に応じてパンクチャする順番を決定する Sorting アルゴリズムから成る．

まず，Grouping アルゴリズムを実行することで全ての変数ノードが復元レベルによって $G_0, G_1, \ldots, G_{k_{\max}}$ に分類される．ここで G_k, $k = 0, 1, \ldots, k_{\max}$, は k-SR ノードの集合を表す．また $k_{\max}$ を最大復元レベルと呼び，BP 復号法を $k_{\max}$ 回繰り返すことで全てのパンクチャド変数ノードを復元できる．Grouping アルゴリズムの詳細な手順の説明は省略するが，このアルゴリズムはできるだけ復元レベルの小さいパンクチャド変数ノードの数が多くなるように選択していく．すなわち，変数ノードの次数とチェックノードの次数が小さいノード同士をパンクチャド変数ノード，Survived チェックノードとして選択していく．この際，選択する候補となる変数ノードが複数ある場合は候補である各変数ノードの誤り率を示す評価関数が最小の値をもつ変数ノードを選択しており，それでも一意に決定されない場合には無作為に選択している．

Sorting アルゴリズムは，Grouping アルゴリズムを実行して得られたパンクチャド変数ノードの集合 $G_1, G_2, \ldots, G_{k_{\max}}$ の中からパンクチャするノードの順番を決定するアルゴリズムである．パンクチャを行わないときの符号化率を R_0 と表し，$G_1, G_2, \ldots, G_{k_{\max}}$ に含まれる全てのノードをパンクチャしたときに得られる最大符号化率を $R_{\max}$ と表す．パンクチャする順番は，$G_1, G_2, \ldots, G_{k_{\max}}$ の順にし，各 G_k の中では多くの Survived チェックノードと接続しているノードから順に選択する．なお，そのような候補が複数ある場合には，次数が小さいノードの順番とし，それでも一意に決定されない場合は候補となるノードの中から無作為に選択する．まず，目的の符号化率 R ($R_0 \leq R \leq R_{\max}$) に対して必要なパンクチャド変数ノードの数 T を計算する．次に T に達するまで Sorting アルゴリズムによって決められた順番にしたがってパンクチャする．

このように復元レベルの小さい変数ノードから順にパンクチャすることで，全てのパンクチャド変数ノードを復元するまでに要する繰り返し回数を少なくすることができる．

2.3 ピボット探索と置換

$\tilde{P}$ 個の連続した変数ノード全てをパンクチャしても復元可能な場合，長さ $\tilde{P}$ のパンクチャ候補とする．このパンクチャ候補の両端のいずれかのノードを候補に加えて長さ $\tilde{P}+1$ に伸ばしたときに復元不可能であるとする．このときピボットを以下のように定義する．

定義 3 復元不可能な長さ $\tilde{P}+1$ のパンクチャ候補のうちある 1 つの変数ノードを候補から除外すると，長さ $\tilde{P}$ のパンクチャ候補が復元可能となるとき，このような変数ノードをピボットと呼ぶ． □

[ピボット探索法]

step1) 適当な長さ $\tilde{P}$ のパンクチャ候補の両端の変数ノードをピボット集合として初期化する．

step2) ピボット集合の中にあるそれぞれのピボットに対し，隣接するチェックノードと長さ $\tilde{P}$ のパンクチャ候補内の変数ノードとの接続数が 1 のとき，それらの変数ノードをピボット集合に追加する．

step3) step2) で新しいピボットが発見されなかった場合，ピボット集合を出力してアルゴリズム終了する．それ以外の場合，step2) へ戻る． □

図 1 にピボット探索法を実行して得られたピボットの例を示す．

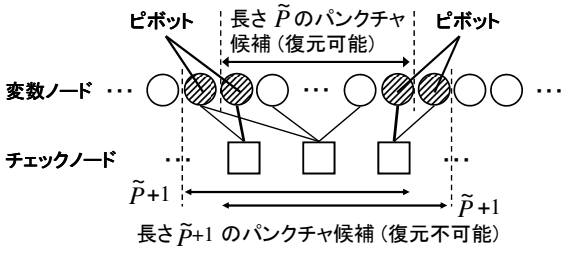

図 1: タナーグラフとピボット

図 1 より，両端のピボットから隣接するチェックノードと長さ $\tilde{P}$ のパンクチャ候補内の変数ノードとの接続数が 1 である変数ノードを順々にたどっていくことで全てのピボットを探索できることがわかる．ピボット探索をしてピボットとパンクチャ候補でない他の変数ノードの置換を繰り返すことで，効率的にパンクチャ候補の数を増やし，符号化率の高い符号を構成することができる．しかし，元々は長いバースト消失を訂正できる符号を構成するために開発された手法 [9] であるため，RCP-LDPC 符号の復号性能に影響する復元レベルを考慮していない．そのため復元レベルが大きい RCP-LDPC 符号を構成してしまう場合がある．

3 提案するパンクチャドビットの選択法

3.1 提案手法の着眼点

Ha らの手法では，優れた復号性能をもつが高い符号化率を達成できないという問題がある．また，ピボットの探索と置換を繰り返す手法は高い符号化率を達成できるが，復元レベルを考慮せずにパンクチャドビットを選択をするため，復号性能が悪くなる可能性がある．そこで提案手法では，Ha らの手法により得られた RCP-LDPC 符号に対して検査行列の構造を利用して効果的に追加のパンクチャドビットを選択し，復元レベルを小さく抑えつつ高い符号化率までパンクチャ可能な RCP-LDPC 符号を構成する手法を提案する．

Grouping アルゴリズムの結果をもとに，検査行列の行と列を適当に置換すると図 2 ようになる．ここで $I^{(k)}$ $(k = 0, 1, \ldots, k_{\max})$ は k-SR ノードと k Survived チェックノードから成る単位行列，A_k は k-SR ノードと $k+1$ Survived チェックノードから成る各行に少なくとも 1 を 1 つもつ行列，B_k は任意の行列を表している．ただし，k Sur は k Survived チェックノードのことであり，$I^{(0)}$, $A_{k_{\max}}$, 0 Sur は存在しない．またパンクチャ候補は $G_1, G_2, \ldots, G_{k_{\max}}$ に対応している．

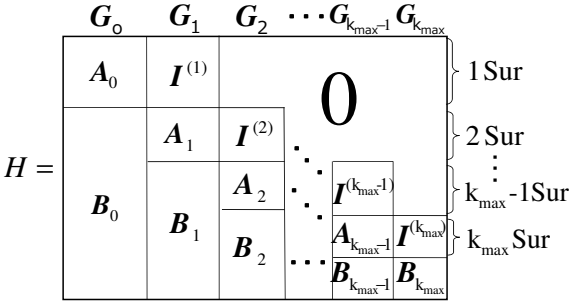

図 2: Grouping アルゴリズムを実行後の検査行列 H

図 2 より，Ha らの手法ではパンクチャド変数ノードを選択する際に，既に選択したパンクチャド変数ノードの復元レベルが大きくならないようにできるだけ大きなサイズの下三角行列を構成していることがわかる．Ha らの手法で構成できない高い符号化率の符号を構成するためには，G_0 からさらに変数ノードを選択し，パンクチャ候補 $G_1, G_2, \ldots, G_{k_{\max}}$ に加える必要がある．このとき 1-SR, 2-SR, ..., $k_{\max}$-SR のチェックノードがより高い復元レベルの Survived チェックノードになり，符号全体としての復元レベルも大きくなってしまう可能性がある．その後，新たに行と列を適当に置換して図 2 のような下三角行列を再び構成したとき，最大復元レベル $k_{\max}$ ができるだけ大きくならないようなパンクチャド変数ノードを選択することが重要であり，なおかつ復元レベルの小さい変数ノードの数を多くすることで復号性能への影響も軽微にできると考えられる．以上のことから，提案手法では以下の 2 つの着眼点に注目しており，それを実現するアルゴリズムを次節で示す．

着眼点 1) G_1 に多くの変数ノード割り当てることで行列 A_1 のサイズが大きくなり，行列 $I^{(2)}$ のサイズが大きくなることが期待できる．これを順々に $G_{k_{\max}}$ まで考慮すると平均的に最大復元レベル $k_{\max}$ を小さくすることが期待できる．

着眼点 2) 変数ノードを選択する際，最大復元レベルが同じになる候補が複数ある場合があるので，復号性能に影響を与えると考えられる評価関数を導入してノードを選択する．

3.2 提案手法

着眼点 1）より，最大復元レベル $k_{\max}$ ができるだけ大きくならないように，まず G_1 に多くの変数ノードを割り当てることを考える．ここで行列 A_0 に注目し，G_0 から選択するある変数ノードと 1 Survived チェックノードを接続する枝の数 (接続数) が多いほど，この変数ノードをパンクチャ候補に加えて図 2 のような下三角行列を再び構成すると 1 Survived チェックノードの数が少なくなり，結果として G_1 に割り当てられる変数ノードが少なくなると予想される．そこでパンクチャ候補 $G_1, G_2, \ldots, G_{k_{\max}}$ に加える変数ノードを G_0 から選択する際に，G_0 内の各変数ノードが 1 Survived チェックノードに接続される数を求め，その値が最小となる変数ノードの中から $k_{\max}$ ができるだけ大きくならないものを選択する．

着眼点 2）より，着眼点 1) で候補となる変数ノードが複数ある場合，それぞれの変数ノードをパンクチャ候補に加えた場合の評価関数を次式のように定義する．

$$S = \sum_{i=1}^{k_{\max}} \{i \times (\text{復元レベル } i \text{ の変数ノードの数})\}. \quad (2)$$

(2) 式をコスト関数 S と呼び，全てのパンクチャド変数ノードが復元されるまでに要する繰り返し回数の総和を定量的に評価するものである．最大復元レベルが同じ場合，(2) 式の値が小さい変数ノードを G_0 から選択することで，優れた復号性能をもつと期待できる．またコスト関数 S が小さいと，着眼点 1) で目指す G_1 のサイズも大きく保持されると期待できる．コスト関数を導入することで，G_0 から選択する変数ノードの候補が複数ある場合に，無作為に選択する方法よりも性能の向上を図る．

以上の考え方により，提案するパンクチャドビットの選択法のアルゴリズムを以下に示す．

[提案アルゴリズム]

step1) LDPC 符号の検査行列に対して Ha らの復元レベルに基づくパンクチャドビットの選択法を実行する．

step2) 検査行列の行と列を適当に置換し，図 2 のように下三角行列を構成する．

step3) G_0 内の各変数ノードに対し，1 Survived チェックノードとの接続数を計算し，接続数が最小の値である変数ノードの集合を得る．この集合内で，パンクチャ候補に加えた場合に，復元可能となる変数ノードが 1 つも見つからないときはアルゴリズムを終了する．最小の値をもつ復元可能な変数ノードが 1 つ存在する場合は，その変数ノードをパンクチャ候補 $G_1, G_2, \ldots, G_{k_{\max}}$ に加えて step2) へ戻る．それ以外の場合は step4) へ行く．

step4) step3) で探索したそれぞれの変数ノードに対し，パンクチャ候補に加えた場合の $k_{\max}$ をそれぞれ求め，最小の値をもつ変数ノードが 1 つ存在する場合は，その変数ノードをパンクチャ候補に加えて step2) へ戻る．それ以外の場合は step5) へ行く．

step5) step4) で探索したそれぞれの変数ノードに対し，パンクチャ候補に加えた場合の S を (2) 式からそれぞれ計算する．最小の値をもつ変数ノードが 1 つ存在する場合は，その変数ノードをパンクチャ候補に加えて step2) へ戻る．それ以外の場合はその変数ノードの中から無作為に選択しパンクチャ候補に加えて step2) へ戻る． □

上記のアルゴリズムを実行して得られた復元可能なパンクチャ候補に含まれる変数ノードを，目的の符号化率 R に応じてパンクチャする．パンクチャする順番は，Ha らの手法によって選んだパンクチャ候補に Sorting アルゴリズムを実行して得られた順とし，それ以降は選択された順にパンクチャする．このようにすることで提案手法では，Ha らの手法で構成できる符号化率までは Ha らの手法と同じ復号性能をもち，それより高い符号化率でも優れた復号性能をもつと考えられる．

4 シミュレーション結果と考察

4.1 シミュレーション条件

提案したパンクチャドビットの選択法の有効性を検証するために，計算機シミュレーションによる実験を行った．実験に使用した符号は確率的に構成した符号長 $N = 1000, 2000$, 符号化率 $R_0 = 0.5$ のそれぞれ正則・非正則 LDPC 符号で，非正則 LDPC 符号は以下の次数分布を用いた[2].

$$\lambda(x) = 0.076923x + 0.692308x^2 + 0.230769x^5,$$
$$\rho(x) = 0.461538x^5 + 0.538462x^6.$$

提案手法 (提案) の比較対象として 2 つの方法を用いた．両手法ともまず Ha らの手法 [4] によってパンクチャ候補を選ぶ．その後に追加として，(1) ランダムにパンクチャドビットを選択する方法を「ランダム手法」(ランダム)，(2) 復元レベルを考慮せず復元可能な長さが大きくなるようにピボット置換を繰り返して選択する方法を「ピボット手法」(ピボット) とした．なお (2) の手法では，(1) の手法でパンクチャ候補が増やせなくなった後，ピボット探索を行って候補の数を増やすものとした．通信路には AWGN 通信路を仮定し，各手法に対して符号語を 10^6 回送信するか，復号が 50 回失敗するまで実験を行った．復号法には BP 復号法を用い，提案手法，ピボット手法，ランダム手法それぞれでパンクチャドビットを選択し，符号化率 $R = 0.80, 0.85, 0.90, 0.95$ の符号を構成した．パンクチャする順番は全ての手法で，Ha らの手法によって選んだパンクチャ候補に Sorting アルゴリズムを実行して得られた順とし，それ以降は選択された順にパンクチャする．なお Ha らの手法では符号化率 $R = 0.78$ までの符号を構成することができ，実験に用いた 3 つの手法は $R = 0.78$ までは Ha らの手法と同じ復号性能をもつ．また，このときの最大復元レベルは 3 である．

[2] 符号化率 $R = 0.50$ あたりの符号が性能が良いので，これをベースとした．また以下の次数分布を用いることにより，復号性能に優れた非正則 LDPC 符号を得られることが知られている．

4.2 符号化率が可変な範囲と最大復元レベル

各手法で得られた RCP-LDPC 符号の最大符号化率と最大復元レベルを表 1〜4 に示す．

表 1: 非正則 LDPC 符号，$N = 1000$

	提案	ピボット	ランダム
最大パンクチャ数 $T_{\max}$	481	481	433
最大符号化率 $R_{\max}$	0.9634	0.9634	0.8818
$T_{\max}$ のときの最大復元レベル	33	106	27

表 2: 非正則 LDPC 符号，$N = 2000$

	提案	ピボット	ランダム
最大パンクチャ数 $T_{\max}$	964	949	876
最大符号化率 $R_{\max}$	0.9652	0.9514	0.8897
$T_{\max}$ のときの最大復元レベル	44	118	42

表 3: 正則 LDPC 符号，$N = 1000$

	提案	ピボット	ランダム
最大パンクチャ数 $T_{\max}$	473	477	440
最大符号化率 $R_{\max}$	0.9487	0.9561	0.8928
$T_{\max}$ のときの最大復元レベル	27	112	35

表 4: 正則 LDPC 符号，$N = 2000$

	提案	ピボット	ランダム
最大パンクチャ数 $T_{\max}$	947	941	852
最大符号化率 $R_{\max}$	0.9496	0.9442	0.8711
$T_{\max}$ のときの最大復元レベル	35	103	48

表 1〜4 の結果から各手法で得られた最大符号化率を比較すると，提案手法とピボット手法が同等の値を示し，ランダム手法よりも大きくなっていることがわかる．

まず，提案手法とランダム手法を比較した場合，達成できる最大符号化率は提案手法の方が大きいことから，提案手法の方が符号化率が可変な範囲が大きいことがわかる．次に，提案手法とピボット手法を比較した場合，達成できる最大符号化率の値は同等であるが最大復元レベルは提案手法の方が大幅に小さくなっている．そのため提案手法は他の手法よりも復号性能が優れていると考えられる．以上の結果から，提案手法は符号化率が可変な範囲を大きくしつつ，復号性能に影響を与える最大復元レベルを小さくできたといえる．また，正則と非正則の符号を比較した場合，非正則 LDPC 符号の方が最大符号化率 $R_{\max}$ が大きく，符号化率の可変な範囲が大きいことがわかる．これは一般的に非正則 LDPC 符号の方が優れた符号を構成しやすい傾向があるためだと考えられる．

4.3 実験結果と考察

(1) 復号性能

符号化率を調整して 3 つの手法の性能を比較した復号結果を図 3, 4 に示す．ここで縦軸は復号後のビット誤り率 (BER) を表し，横軸は AWGN 通信路の 1 ビット当たりの信号対雑音比 (SN 比) [dB] を表す．図において SN 比が上がっても BER が減少してない場合は，全てのパンクチャドビットを復号できていないことを意味しており，$R = 0.90, 0.95$ のランダム手法がこれに値する．また $R = 0.80, 0.85$ におけるピボット手法とランダム手法のパンクチャドビットは同じであるため，ランダム手法としてまとめた[3]．

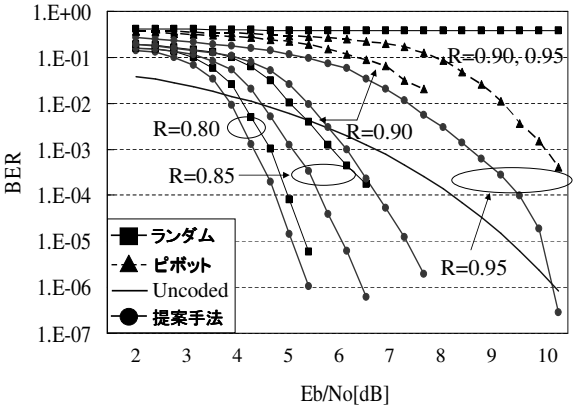

図 3: 復号結果 (非正則，$N = 1000$)

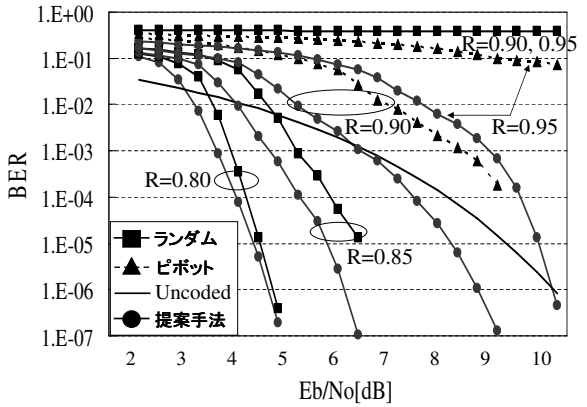

図 4: 復号結果 (正則，$N = 2000$)

図 3, 4 より，実験を行った符号化率においては提案手法がピボット手法よりも優れた復号性能を示しており，特に符号化率が高いところではその差が顕著であることがわかる．また提案手法と Uncoded の誤り確率を比較した場合，$R = 0.80, 0.85, 0.90$ において SN 比が高い領域で提案手法が優れた復号性能を示している．このことから $R = 0.80, 0.85, 0.90$ においては，実装するにあたり特に提案手法の有用性が高いと考えられる．最大復元レベル，コスト関数 S を考慮したことが性能向上に効果的であったと考えられ，最大復元レベル，コスト関数 S それぞれに関する考察は以降の (2), (3) において行う．

[3] ピボット手法は，ランダム手法でパンクチャ候補を増やせなくなった $R = 0.8818$ 以上で追加するパンクチャドビットを探索するからである．

(2) 復元レベル

各手法のパンクチャ数 T ごとの最大復元レベルをまとめた表 5 を示す．

表 5: 各手法のパンクチャ数 T ごとの最大復元レベル．–は復元不可能であることを表す．(非正則，$N = 1000$)

T	符号化率	提案	ピボット	ランダム
375	0.80	4	5	5
411	0.85	5	11	11
444	0.90	10	51	–
450	0.91	10	60	–
456	0.92	13	67	–
462	0.93	14	78	–
468	0.94	20	82	–
473	0.95	21	86	–
481	0.9634	33	106	–

表 5 の結果より，全ての符号化率において提案手法の最大復元レベルがピボット手法，ランダム手法よりも小さくなっていることがわかる．符号化率が高くなるほど提案手法とピボット手法との最大復元レベルの差が大きいことから，高い符号化率における提案手法の有効性が示されたと言える．また提案手法とピボット手法では，高い符号化率において，符号化率のわずかな違いで最大復元レベルが急激に増加することがわかる．したがって，達成可能な最大符号化率の符号よりも少し小さい符号化率の符号までを可変領域として用いることが現実的と考えられる．

(3) コスト関数 S

提案手法の (2) 式において導入したコスト関数 S の有効性を確認するために，コスト関数 S を考慮せず最大復元レベルのみを考慮した手法を比較手法 (比較) として実験を行った．符号化率 $R = 0.80, 0.85, 0.90, 0.95$ における提案手法と比較手法の復号結果を図 5 に示す．また，提案手法と比較手法の各符号化率ごとの最大復元レベルを表 6 に示す．

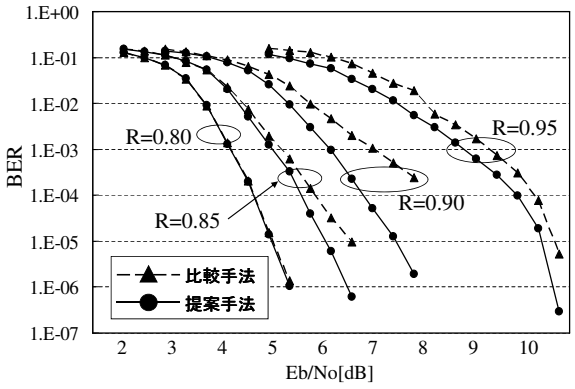

図 5: 復号結果（非正則，$N = 1000$）

図 5 の結果から，実験した全ての符号化率において提案手法の復号性能の方が優れていることがわかる．また，表 6 から各符号化率において提案手法の復元レベルは比較手法と同じかもしくは小さいことがわかる．これにより，最大復元レベルを抑えつつ，コスト関数 S を考慮することでパンクチャ

表 6: 符号化率ごとの最大復元レベル（非正則，$N = 1000$）

符号化率	提案	比較
0.80	4	4
0.85	5	10
0.90	10	14
0.95	21	36

ドビット全体の復元レベルが小さく抑えられ，最大復元レベルが同じ場合でも優れた復号性能を示すことがわかる．

5 まとめと今後の課題

本研究では，著者らが提案している手法に対して様々な観点から評価を行った．提案している手法では高い符号化率までパンクチャできる復号性能に優れた RCP-LDPC 符号を構成するために，検査行列の構造を利用し復元レベルが大きくならないようにビットを選択している．またパンクチャドビットを選択する際，候補となるビットが複数存在する場合も多いため，ビットごとにコスト関数 S を導入して無作為に選択するよりも性能の向上を図っている．

計算機シミュレーションによる実験結果から，提案したパンクチャドビットの選択法は復元レベルを考慮しないピボットの置換手法とほぼ同程度の符号化率の符号を構成できると同時に，全てのパンクチャドビットの復元に要する BP 復号法の繰り返し回数が少ないという特徴を併せ持つ．そのため提案手法は，ピボットの置換手法，ランダムに選択する手法よりも優れた復号性能をもつことを示した．

本研究では低い符号化率の領域において Ha らの手法でパンクチャドビットを選ぶことを前提としていたが，低い領域で選ぶパンクチャドビットによって高い符号化率での性能が大きく左右されることがわかっている．Ha らの手法でもコスト関数を導入しているが，最初の段階でのビット選択において，候補となる無数のビットの中から確率的に選択していることが多い．今後の課題としては元の符号からパンクチャドビットを選ぶ際，Ha らの手法とは異なるコスト関数を導入してさらなる復号性能の向上を図ることが考えられる．

参考文献

[1] R. G. Gallager, *Low density parity check codes*, MIT Press, 1963.

[2] D. J. C. MacKay, “Good error-correcting codes based on very sparse matrices,” *IEEE Trans. Inform. Theory*, Vol. 45, No. 2, pp. 399–431，March 1999.

[3] J. Ha, J. Kim, and S. W. McLaughlin, “Rate-compatible puncturing of low-density parity-check codes,” *IEEE Trans. Inform. Theory*, Vol. IT-50, No. 11, pp. 2824–2836, Nov. 2004.

[4] J. Ha, J. Kim, D. Klinc, and S. W. McLaughlin, “Rate-compatible punctured low-density parity-check codes with short block lengths,” *IEEE Trans. Inform. Theory*, Vol. 52, No. 2, pp. 728–738, Feb. 2006.

[5] 長田佳史, 寺本賢一, 細谷剛, 後藤正幸, “高符号化率までパンクチャ可能な LDPC 符号に関する一考察,” 電子情報通信学会技術研究報告 *IT*, Vol. 110, No. 137, pp. 95–100, 2010 年 7 月.

[6] J. Kwon, D. Klinc, J. Ha, and W. McLaughlin, “Fast decoding of rate-compatible punctured low-density parity-check codes,” *Proc. 2007 IEEE International Symposium on Information Theory*, pp. 216–220, Nice France, June 2007.

[7] B. N. Vellambi and F. Fekri, “Finite-length rate-compatible low-density parity-check codes : A novel puncturing scheme,” *IEEE Trans. Commun.*, Vol. 57, No. 2, pp. 297–301, Feb. 2009.

[8] J. Kim, A. Ramamoorthy, and S. W. McLaughlin, “The design of efficiently-encodable rate-compatible LDPC codes,” *IEEE Trans. Commun*, Vol. 57, No. 2, pp. 365–375, Feb. 2009.

[9] E. Paolini and M. Chiani, “Construction of near-optimum burst erasure low-density parity-check codes,” *IEEE Trans. Commun.*, Vol. 57, No. 5, pp. 1320–1328, May 2009.

[10] M. El-Khamy, J. Hou, and N. Bhushan, “Design of rate-compatible structured LDPC codes for hybrid ARQ applications,” *IEEE Journal on selected areas in Commun.*, Vol. 27, No. 6, pp. 965–973, Aug. 2009.