低密度パリティ検査符号 (LDPC 符号)

—Robert G. Gallager 先生の 2020 年日本国際賞受賞に寄せて

Low-Density Parity-Check Codes

—Celebrating Receipt of Japan Prize by Prof. Robert G. Gallager

内川浩典 Hironori UCHIKAWA

アブストラクト 2020 年の日本国際賞をマサチューセッツ工科大学名誉教授である Robert G. Gallager が受賞した。受賞の対象となった低密度パリティ検査符号 (LDPC 符号) は、Gallager が 1960 年に彼の博士論文で提案した誤り訂正符号である。30 年以上という長い歳月を経て評価された LDPC 符号は、またたく間に我々に身近な情報機器にも搭載されるようになり、現在では我々の生活に欠かせない技術となっている。本稿ではこの LDPC 符号について解説する。特に確率推論の観点から最適な復号アルゴリズムが LDPC 符号により実行可能な形で導かれることを紹介する。更に実用的な LDPC 符号をどのようにして構成するのか、その構成方法を紹介する。

キーワード LDPC 符号, 最大事後確率復号, sum-product 復号, 原始グラフ, 準巡回 LDPC 符号, 差分進化法

Abstract Robert G. Gallager, Professor Emeritus at Massachusetts Institute of Technology, was awarded the 2020 Japan Prize. The low-density parity-check code (LDPC code), the award's subject, is an error correction code proposed by Professor, Gallager in his doctoral thesis in 1960. It has now become an essential fundamental technology that supports our modern digital life. In this article, we describe the LDPC codes that can derive the optimal decoding algorithm in probabilistic inference. Furthermore, in this article we also introduce a practical LDPC code construction.

Key words LDPC codes, Maximum a posteriori decoding, Sum-product decoding, Protograph, Quasi-cyclic LDPC codes, Differential evolution

1. ま え が き

現在、世の中はデータ爆発時代といわれている。老若男女を問わず、誰もがスマートフォンやタブレットといった情報機器でデジタルデータのやりとり、蓄積を行っている。そしてそのような情報機器では、蓄積されたデジタルデータを読み出す際や伝送データを受け取る際、データに生じた誤りを訂正するため誤り訂正符号が用いられている。例えばスマートフォンで撮影した写真が消えることなくいつでも閲覧できるのも、誤り訂正符号のおかげである^(注1)。

Shannon が誤りなく通信が可能な通信速度の限界を示して以降⁽²⁾、この理論限界を達成する誤り訂正符号の確立が待ち望まれていた。驚くべきことに、Shannon が理論限界を示した後 20 年も経たないうちに、Gallager は彼の博士論文⁽³⁾の中で、理論限界に迫る訂正能力を (今となっては) 実現可能な計算量で実行

できる誤り訂正符号、低密度パリティ検査符号 (Low-Density Parity-Check Codes: 以下 LDPC 符号) を発明していた。彼が発明した確率的な復号アルゴリズムは、残念なことに当時の計算機ではその有効性を確認できなかったため、1990 年代後半に MacKay によって再発見されるまで⁽⁴⁾、その高い訂正能力が世間に認知されることはなかった。

2000 年に入ってから以降、Richardson と Urbanke によって Shannon の理論限界をほぼ達成する LDPC 符号の構成法⁽⁵⁾、⁽⁶⁾が発明されたことを皮切りに、符号の構成法や復号アルゴリズムに関する数多くの研究がなされ、LDPC 符号は符号理論分野における一大ブームとなった。

また産業界もその訂正能力に注目し、デジタル放送や無線 LAN といった通信システムが次々に LDPC 符号を採用し、今やハードディスクドライブやソリッドステートドライブといったストレージシステムにも採用されるなど、我々に身近な情報機器にも搭載されるようになってきている。

このような背景から Gallager による LDPC 符号の発明は、科学技術の進歩にとどまらず、私たちの暮らしを豊かにする社会的貢献度が大きいことが評価され、彼が LDPC 符号についての博士論文をまとめてからちょうど 60 年経った 2020 年に、Gallager は日本国際賞を受賞することになった。

本稿ではこの LDPC 符号について解説する。前半では、確率推論の観点から最適な復号アルゴリズムが LDPC 符号により実行可能な形で導かれることを紹介する。後半では、少し高度な話題になるが実用的な LDPC 符号の構成方法を紹介する。

(注1)：本稿は著者個人の見解に基づくものであり、所属組織・部門見解を代表するものではありません。また『進化する符号理論』(日本評論社, 2016)⁽¹⁾の著述を出版社の許諾のもと一部転載しています (一部の変更も含む)。

内川浩典 正員：シニア会員 キオクシア株式会社メモリ技術研究所
E-mail hironori.uchikawa@kioxia.com
Hironori UCHIKAWA, Senior Member (Institute of Memory Technology R&D, Kioxia Corporation).
電子情報通信学会 基礎・境界サイエンス
Fundamentals Review Vol.14 No.3 pp.217–228 2021 年 1 月
©電子情報通信学会 2021

図 1 通信システムモデル

2. 通信システムモデルと誤り訂正符号

誤り訂正符号を議論する際の通信システムモデルは図 1 のように表される。まず、情報を表す長さ k のメッセージ $\mathbf{m} = m_1 m_2 \cdots m_k$ は、通信路での誤りに耐えられるよう、符号化処理によって長さ n の符号語 $\mathbf{x} = x_1 x_2 \cdots x_n$ へと変換される。ただし $n > k$ である。符号語 $\mathbf{x}$ は、通信路において誤りが加わり受信語 $\mathbf{y} = y_1 y_2 \cdots y_n$ となる。ここで通信路は、送信された符号語 $\mathbf{x}$ に対する条件付き確率 $P_{Y|X}(\mathbf{y}|\mathbf{x})$ でモデル化される。受信語 $\mathbf{y}$ は復号処理により、もとの符号語の推定語 $\hat{\mathbf{x}}$ 若しくはメッセージの推定語 $\hat{\mathbf{m}}$ へと変換される^(注2)。なお本稿では議論を簡単にするため、メッセージのシンボル m_i ($1 \leq i \leq k$)、符号語のシンボル x_i ($1 \leq i \leq n$) 及び受信語のシンボル y_i ($1 \leq i \leq n$) はどちらも 2 元体 $\mathbb{F}_2$ の元 (つまり 0 か 1) のいずれかをとり、通信路はシンボルごとに独立かつ同一の条件付き確率

$$P_{Y|X}(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^n P_{Y_i|X_i}(y_i|x_i),$$

ただし

$$P_{Y_i|X_i}(y_i|x_i) = \begin{cases} 1-p & x_i = y_i \\ p & x_i \neq y_i \end{cases} \quad (1)$$

でモデル化された 2 元対称通信路とする。 p を通信路誤り率と呼び、通信路でシンボルが誤る確率を表す。

通信の伝送速度を $R = k/n$ と定義したとき、通信路モデルから導出される通信路容量 C を超えない範囲 $R < C$ で、復号誤り率を任意に小さくできる誤り訂正符号が存在することが知られている⁽²⁾。そして誤り訂正符号に携わる技術者にとっては、限りなく C に近い伝送速度 R を計算量的に実行可能な符号化復号処理で実現することが、大きな目標となっている。

3. 確率推論による復号と LDPC 符号

3.1 確率推論による復号

誤り訂正符号の復号処理とは、通信路で誤りが生じた受信語 $\mathbf{y}$ から、もっともらしい符号語 $\mathbf{x}$ を推定する推論問題と考えることができる。そしてこの「もっともらしさ」を定量的に扱う道具として、確率を用いる。

(注2)：通信システムとしてはもとのメッセージの推定語 $\hat{\mathbf{m}}$ が得られるまでを通信システムモデルとすべきだが、メッセージ $\mathbf{m}$ と符号語 $\mathbf{x}$ とは 1 対 1 対応の関係があるため、送信された符号語の推定語 $\hat{\mathbf{x}}$ が得られるまでを通信システムモデルとすることが多い。

いま、送信者が長さ n の符号語の集合である符号 $\mathcal{C}$ から符号語 $\mathbf{x} = x_1 x_2 \cdots x_n \in \mathcal{C}$ を一様を選んで送信し、通信路を介して受信者が受信語 $\mathbf{y} = y_1 y_2 \cdots y_n$ を受信することとする。このとき、符号語シンボル x_i の誤り率を最も小さくする復号アルゴリズムは次のように与えられる。

$$\hat{x}_i = \operatorname{argmax}_{x_i \in \{0,1\}} P_{X|Y}(x_i|\mathbf{y}) \quad (2)$$

ただし $\operatorname{argmax}_{x_i \in \{0,1\}}$ は右項を最大にする引数 x_i を返す関数で、 $P_{X|Y}(x_i = 0|\mathbf{y})$ と $P_{X|Y}(x_i = 1|\mathbf{y})$ とを比較し、確率値の大きい x_i を推定シンボル $\hat{x}_i$ とする。事後確率と呼ばれる $P_{X|Y}(x_i|\mathbf{y})$ を最大化するシンボルを推定シンボルとすることから、この復号アルゴリズムは最大事後確率復号^(注3)と呼ばれる。

式 (2) を通信路の条件付き確率 $P_{Y|X}(y|x)$ で計算できる形に式変形すると、次のようになる。

$$\begin{aligned} \hat{x}_i &= \operatorname{argmax}_{x_i \in \{0,1\}} P_{X|Y}(x_i|\mathbf{y}) \\ &= \operatorname{argmax}_{x_i \in \{0,1\}} \sum_{\mathbf{x}} P_{X|Y}(\mathbf{x}|\mathbf{y}) \end{aligned} \quad (3)$$

$$= \operatorname{argmax}_{x_i \in \{0,1\}} \sum_{\mathbf{x}} \frac{P_{Y|X}(\mathbf{y}|\mathbf{x}) P_{\mathbf{X}}(\mathbf{x})}{P_{\mathbf{Y}}(\mathbf{y})} \quad (4)$$

$$= \operatorname{argmax}_{x_i \in \{0,1\}} \sum_{\mathbf{x}} P_{Y|X}(\mathbf{y}|\mathbf{x}) P_{\mathbf{X}}(\mathbf{x}) \quad (5)$$

$$= \operatorname{argmax}_{x_i \in \{0,1\}} \sum_{\mathbf{x}} \left(\prod_{j=1}^n P_{Y_j|X_j}(y_j|x_j) \right) \mathbb{I}[\mathbf{x} \in \mathcal{C}] \quad (6)$$

なお、 $\sum_{\mathbf{x}}$ は x_i を除く全てのシンボルの、全ての値に対する総和

記号で、例えば $\sum_{\mathbf{x}} = \sum_{x_2 \in \{0,1\}} \sum_{x_3 \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}}$

を表す。式 (3) は周辺確率の計算式^(注4)から得られ、式 (3) から式 (4) への変形は

$$\begin{aligned} P_{Y,\mathbf{X}}(\mathbf{y},\mathbf{x}) &= P_{X|Y}(\mathbf{x}|\mathbf{y}) P_{\mathbf{Y}}(\mathbf{y}) \\ &= P_{Y|X}(\mathbf{y}|\mathbf{x}) P_{\mathbf{X}}(\mathbf{x}) \end{aligned}$$

というベイズ則から得られる。そして $P_{\mathbf{Y}}(\mathbf{y})$ が $\operatorname{argmax}_{x_i \in \{0,1\}}$ の操作に寄与しないことから、式 (5) が得られる。更に各シンボルの通信路誤り率が独立なことから、各符号語が一様に送信される

$$P_{\mathbf{X}}(\mathbf{x}) = \begin{cases} 1/|\mathcal{C}| & (\mathbf{x} \in \mathcal{C}) \\ 0 & (\text{otherwise}) \end{cases} = \frac{1}{|\mathcal{C}|} \mathbb{I}[\mathbf{x} \in \mathcal{C}],$$

ことから式 (6) を得た。ただし $\mathbb{I}[\text{条件}]$ は指示関数で、条件を満たすときには 1 を返し、そうでない場合は 0 を返す。

(注3)：送信された符号語 $\mathbf{X}$ の事後確率を最大化する最大事後確率復号と区別するため、シンボル最大事後確率復号や最大事後周辺確率復号と呼ばれることもある。

(注4)：多変数の同時確率から計算される単一変数の確率を、周辺確率と呼ぶ。例えば 2 変数の同時確率 $P_{AB}(a,b)$ が与えられたとき、確率変数 A の周辺確率は $P_A(a) = \sum_{b \in \mathcal{B}} P_{AB}(a,b)$ により得られる。ただし $\mathcal{B}$ は確率変数 B の定義域を表す。

式 (6) であれば、通信路の条件付き確率から直接計算できるものの、 $\sum_{\sim x_i}$ で総和をとる項数が符号の長さ n に対して指数的に増加するため、一定以上の長さの符号に対してこの計算をこのまま実行することは現実的ではない。では効率的に計算するには、どのようにすればよいだろうか？

3.2 分配則による演算数の削減

前節の式 (6) を見ると分かるように、事後確率の計算は積和演算の形をとる。もし、和の項に共通する因子を和記号の外にくくり出す、つまり分配則が適用できれば、演算数を削減できる (図 2)。

分配則

$$ax + ay = a(x + y)$$

乗算2回	乗算1回
加算1回	加算1回

図 2 分配則により、乗算数を一つ減らせる例

分配則により事後確率の計算式が簡単になることを示すため、以下では $\mathbb{F}_2$ 上のパリティ検査行列

$$\mathbf{H}_1 = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (7)$$

で定義される符号

$$\mathcal{C}_1 = \{\mathbf{x} \in \{0, 1\}^7 | \mathbf{H}_1 \mathbf{x}^T = \mathbf{0}\}$$

を用いて、送信シンボル x_1 に対する事後確率の計算式を導く。

まず、式 (6) より、送信シンボル x_1 に対する事後確率は次のように与えられる。

$$P_{X|Y}(x_1|y) = \sum_{\sim x_1} \mathbb{I}[\mathbf{H}_1 \mathbf{x}^T = \mathbf{0}] \prod_{j=1}^7 P_{Y|X}(y_j|x_j) \quad (8)$$

なお、 $\mathbf{x}^T$ はベクトル $\mathbf{x}$ の転置を表す。ここで、指示関数の条件となっている $\mathbf{H}_1 \mathbf{x}^T = \mathbf{0}$ がパリティ検査行列中の各検査式の充足条件の積で記述できることを利用すると、式 (8) は

$$P_{X|Y}(x_1|y) = \sum_{\sim x_1} \mathbb{I}_1 \mathbb{I}_2 \mathbb{I}_3 \prod_{j=1}^7 P_{Y|X}(y_j|x_j)$$

と変形できる。ただし

$$\mathbb{I}_1 = \mathbb{I}[x_1 + x_4 + x_5 = 0],$$

$$\mathbb{I}_2 = \mathbb{I}[x_1 + x_2 + x_6 = 0],$$

$$\mathbb{I}_3 = \mathbb{I}[x_2 + x_3 + x_7 = 0]$$

で、指示関数の条件はそれぞれ 1 行めから 3 行めのパリティ検査式に対応している。更に、総和に含まれる共通因子と指示関

数の条件に着目すると、次のように変形できる。

$$P_{X|Y}(x_1|y) = P_{Y|X}(y_1|x_1) \sum_{\sim x_1} \mathbb{I}_1 \mathbb{I}_2 \mathbb{I}_3 \prod_{j=2}^7 P_{Y|X}(y_j|x_j) \quad (9)$$

$$= P_{Y|X}(y_1|x_1) \left(\sum_{x_4 x_5} \mathbb{I}_1 \prod_{j' \in \{4, 5\}} P_{Y|X}(y_{j'}|x_{j'}) \right) \times \left(\sum_{x_2 x_3 x_6 x_7} \mathbb{I}_2 \mathbb{I}_3 \prod_{j'' \in \{2, 3, 6, 7\}} P_{Y|X}(y_{j''}|x_{j''}) \right) \quad (10)$$

$$= P_{Y|X}(y_1|x_1) \left(\sum_{x_4 x_5} \mathbb{I}_1 \prod_{j' \in \{4, 5\}} P_{Y|X}(y_{j'}|x_{j'}) \right) \times \left(\sum_{x_2 x_6} \mathbb{I}_2 \prod_{j'' \in \{2, 6\}} P_{Y|X}(y_{j''}|x_{j''}) \right) \times \left(\sum_{x_3 x_7} \mathbb{I}_3 \prod_{j''' \in \{3, 7\}} P_{Y|X}(y_{j'''}|x_{j'''}) \right) \quad (11)$$

なお $\sum_{x_a x_b}$ はシンボル x_a と x_b の全ての要素についての総和記号を表す。例えば $\sum_{x_4 x_5}$ は、 $\sum_{x_4 \in \{0, 1\}} \sum_{x_5 \in \{0, 1\}}$ を表す。

まず、全ての和の項に含まれる $P_{Y|X}(y_1|x_1)$ を和記号の外へくくり出すことで、式 (9) が得られる。次に、検査式 $x_1 + x_4 + x_5 = 0$ で変数の組が決まる $x_4 x_5$ をくくり出し、式 (10) が得られる。最後に、残りの検査式に対応する指示関数 $\mathbb{I}_2, \mathbb{I}_3$ ごとに積和をまとめることで、式 (11) を得た。変形前の式 (8) では、乗算の数が 96、加算の数が 14 必要であったのに対し、変形後の式 (11) では、乗算の数が 20、加算の数が 6 となり、演算数の少ない計算式を得ていることが分かる。この例ではもとの式 (8) の演算数がもともと大きくないため、分配則による演算数削減の恩恵が少なくないように感じるかもしれないが、変形前の式は符号長が長くなると演算数が指数的に増加し、直接計算することが難しくなる。一方、式 (11) は次節以降で述べる sum-product アルゴリズムを用いることで、非常に効率よく計算することができる。

3.3 木と sum-product アルゴリズム

前節では、事後確率の計算式が分配則により指示関数ごとの積和式の積へ簡略化できることを、例を用いて説明した。どのようなパリティ検査行列に対しても、上述したような簡略化が適用できれば好ましいのだが、残念ながらこのような式変形はパリティ検査式を 2 部グラフで表現した際に、そのグラフ構造が木になっている場合のみに限られる。

図 3 に式 (7) のパリティ検査式の 2 部グラフ表現を示す。2 部グラフは、符号語シンボルに対応する変数ノード (○) と、パリティ検査式に対応する検査ノード (□) の 2 種類のノードと、それらを接続するエッジ (辺) から構成される無向グラフである。パリティ検査行列 H の i 行 j 列の要素を $h_{i,j}$ とすると、

図 3 式 (7) のパリティ検査式の 2 部グラフ表現

図 4 符号シンボル x_1 の事後確率を sum-product アルゴリズムで計算する際のメッセージの流れ

$h_{i,j} \neq 0$ のときに 2 部グラフ上の j 番目の変数ノード v_j と i 番目の検査ノード c_i とがエッジ (i, j) により接続される。つまり 2 部グラフの変数ノード v_j は符号シンボル x_j に対応し、検査ノード c_i は i 行めのパリティ検査式に対応する。例として式 (7) のパリティ検査行列 H_1 の 1 列めのベクトルを見てみよう。 $(h_{1,1}, h_{2,1}, h_{3,1})^T = (1, 1, 0)^T$ であることから、符号語シンボル x_1 は 1 行めと 2 行めのパリティ検査式に含まれていることが分かる。一方、図 3 の 2 部グラフでは、符号語シンボル x_1 に対応する変数ノード v_1 が、1 行めと 2 行めのパリティ検査式に対応する検査ノード c_1 と c_2 に接続されている。

グラフ構造が木とは、任意のノード a からその他の任意のノード b への経路が一つしかないということを表し、閉路をもたないグラフを意味する。図 3 を見ると、グラフに閉路がなく、木になっていることが分かるだろう。グラフが木の場合、sum-product アルゴリズムにより組織的に事後確率を計算することができる。

sum-product アルゴリズムでは、事後確率を計算したいノード (図 3 では v_1) をルートノードにし、ルートノードからの経路の末端にある葉ノード (図 3 では $v_2, v_3, \dots, v_7$) からルートノードに向かって、ノードごとに式 (12), (13) の処理を行い、生成したメッセージをルートノードの方向へに送る (図 4)。そしてルートノードは接続している全てのエッジからメッセージを受け取った後、式 (14) の一時推定処理をすることで、事後確率を得る。

変数ノード処理

$$M_{v_j \rightarrow c_i}(x_j) = P_{Y|X}(y_j|x_j) \prod_{i' \in \mathcal{I}(j) \setminus \{i\}} M_{c_{i'} \rightarrow v_j}(x_j) \quad (12)$$

検査ノード処理

$$M_{c_i \rightarrow v_j}(x_j) = \sum_{\sim x_j} \mathbb{I}_i \prod_{j' \in \mathcal{J}(i) \setminus \{j\}} M_{v_{j'} \rightarrow c_i}(x_{j'}) \quad (13)$$

一時推定処理

$$g(x_j) = P_{Y|X}(y_j|x_j) \prod_{i \in \mathcal{I}(j)} M_{c_i \rightarrow v_j}(x_j) \quad (14)$$

なお、 $\mathcal{I}(j)$ は変数ノード v_j に接続する検査ノードのインデックス (シンボル x_j が含まれるパリティ検査式のパリティ検査行列中の行インデックス) の集合を表し、 $\mathcal{J}(i)$ は検査ノード c_i に接続する変数ノードのインデックス (i 行めのパリティ検査式に含まれるシンボル x_j のインデックス) の集合を表す。また $\mathcal{S} \setminus \{i\}$ は、集合 $\mathcal{S}$ から集合 $\{i\}$ を除いた差集合を表す。なお、葉ノードはエッジが 1 本のため $\mathcal{I}(j) \setminus \{i\}$ は空集合となり、検査ノードへ送るメッセージの計算式 (12) 中の、ほかの検査ノードから受信するメッセージの積の項をもたない。例えば変数ノード v_3 が検査ノード c_3 へ送るメッセージは

$$M_{v_3 \rightarrow c_3}(x_3) = P_{Y|X}(y_3|x_3) \quad (15)$$

で計算される。 v_1 以外の変数ノードに対しても、ルートノードを入れ替えることで同様に計算できる。

具体例として、式 (7) のパリティ検査行列 H_1 で定義される符号 $\mathcal{C}_1$ で符号化された符号語が、通信路誤り率 p の BSC を介して受信語 $\mathbf{y} = 1000000$ として受信した際の $P_{X|\mathbf{Y}}(x_1|\mathbf{y})$ を図 4 で示した流れにそって、sum-product アルゴリズムで計算してみよう。

葉ノードである v_3 が検査ノード c_3 へ送るメッセージ $M_{v_3 \rightarrow c_3}(x_3)$ は、式 (1), (15) と $y_3 = 0$ より、

$$M_{v_3 \rightarrow c_3}(x_3) = \begin{cases} 1-p & x_3 = 0 \\ p & x_3 = 1 \end{cases} \quad (16)$$

となる。同様に、葉ノード v_7 が検査ノード c_3 へ送るメッセージも

$$M_{v_7 \rightarrow c_3}(x_7) = \begin{cases} 1-p & x_7 = 0 \\ p & x_7 = 1 \end{cases} \quad (17)$$

となる。

検査ノード c_3 が変数ノード v_2 へ送るメッセージ $M_{c_3 \rightarrow v_2}(x_2)$ は、式 (13) より、

$$M_{c_3 \rightarrow v_2}(x_2 = 0) = M_{v_3 \rightarrow c_3}(x_3 = 0)M_{v_7 \rightarrow c_3}(x_7 = 0) \\ + M_{v_3 \rightarrow c_3}(x_3 = 1)M_{v_7 \rightarrow c_3}(x_7 = 1),$$

$$M_{c_3 \rightarrow v_2}(x_2 = 1) = M_{v_3 \rightarrow c_3}(x_3 = 1)M_{v_7 \rightarrow c_3}(x_7 = 0) \\ + M_{v_3 \rightarrow c_3}(x_3 = 0)M_{v_7 \rightarrow c_3}(x_7 = 1)$$

で計算できる^(注5)。これらの式に式 (16), (17) を代入することで、検査ノード c_3 が変数ノード v_2 へ送るメッセージ

$$M_{c_3 \rightarrow v_2}(x_2) = \begin{cases} p^2 + (1-p)^2 & x_2 = 0 \\ 2p(1-p) & x_2 = 1 \end{cases} \quad (18)$$

を得る。

次に変数ノード v_2 が検査ノード c_2 へ送るメッセージは式 (12), (18) と $y_2 = 0$ より

$$M_{v_2 \rightarrow c_2}(x_2) = P_{Y|X}(y_2|x_2)M_{c_3 \rightarrow v_2}(x_2) \\ = \begin{cases} (1-p)(p^2 + (1-p)^2) & x_2 = 0 \\ 2p^2(1-p) & x_2 = 1 \end{cases} \quad (19)$$

と計算される。変数ノード v_6 は葉ノードであるため、検査ノード c_2 へ送るメッセージは $y_6 = 0$ より

$$M_{v_6 \rightarrow c_2}(x_6) = \begin{cases} 1-p & x_6 = 0 \\ p & x_6 = 1 \end{cases} \quad (20)$$

となる。そして検査ノード c_2 が変数ノード v_1 へ送るメッセージは式 (13) より、

$$M_{c_2 \rightarrow v_1}(x_1 = 0) = M_{v_2 \rightarrow c_2}(x_2 = 0)M_{v_6 \rightarrow c_2}(x_6 = 0) \\ + M_{v_2 \rightarrow c_2}(x_2 = 1)M_{v_6 \rightarrow c_2}(x_6 = 1), \\ M_{c_2 \rightarrow v_1}(x_1 = 1) = M_{v_2 \rightarrow c_2}(x_2 = 1)M_{v_6 \rightarrow c_2}(x_6 = 0) \\ + M_{v_2 \rightarrow c_2}(x_2 = 0)M_{v_6 \rightarrow c_2}(x_6 = 1)$$

となるので、これらの式に式 (19), (20) を代入し

$$M_{c_2 \rightarrow v_1}(x_1 = 0) = (1-p)^2(p^2 + (1-p)^2) + 2p^3(1-p), \quad (21)$$

$$M_{c_2 \rightarrow v_1}(x_1 = 1) = p(1-p)(p^2 + (1-p)^2) + 2p^2(1-p)^2 \quad (22)$$

を得る。

検査ノード c_1 が変数ノード v_1 へ送るメッセージ $M_{c_1 \rightarrow v_1}(x_1)$ は、検査ノード c_3 が変数ノード v_2 へ送るメッセージ $M_{c_3 \rightarrow v_2}(x_2)$ と同様に計算でき、

$$M_{c_1 \rightarrow v_1}(x_1) = \begin{cases} p^2 + (1-p)^2 & x_1 = 0 \\ 2p(1-p) & x_1 = 1 \end{cases} \quad (23)$$

となる。

最後に、式 (14) より、送信シンボル x_1 の事後確率は

$$g(x_1) = P_{Y|X}(y_1|x_1)M_{c_1 \rightarrow v_1}(x_1)M_{c_2 \rightarrow v_1}(x_1) \quad (24)$$

(注5)：これは指示関数 $\mathbb{I}_3$ の条件であるパリティ検査式 $x_2 + x_3 + x_7 = 0$ に対応している。 $x_2 = 0$ となるためには $x_3 = x_7$ つまり x_3, x_7 が 00 若しくは 11 とならなければならない。逆に $x_2 = 1$ となるためには $x_3 \neq x_7$ つまり x_3, x_7 が 10 若しくは 01 となる必要がある。

で計算される。受信シンボル $y_1 = 1$ から

$$P_{Y|X}(y_1|x_1) = \begin{cases} p & x_1 = 0 \\ 1-p & x_1 = 1 \end{cases} \quad (25)$$

となり、式 (24) に式 (21), (22), (23), (25) を代入し、式を展開すると

$$g(x_1 = 0) = -8p^6 + 22p^5 - 26p^4 + 17p^3 - 6p^2 + p,$$

$$g(x_1 = 1) = -2p^5 + 6p^4 - 6p^3 + 2p^2$$

が得られる。これは式 (8) で計算される事後確率 $P_{X|Y}(x_1|y)$ と一致する^(注6)。

3.4 同時更新型 sum-product 復号

復号の際には、全てのシンボル x_j ($1 \leq j \leq n$) に対して事後確率を計算するが、各シンボルに対応する変数ノードをルートノードとして計算を繰り返すことは、シンボル間に共通するメッセージが存在するため、効率がよくない。そこで通常は各ノードの処理を同時に行い、収束するあるいはあらかじめ定めた最大繰り返し回数 K まで繰り返し処理をする同時更新型 sum-product アルゴリズムが用いられる。以下では、同時更新型 sum-product アルゴリズムを使った復号法の手順を紹介する。

同時更新型 sum-product 復号アルゴリズム

手順 1 初期化

K に最大繰り返し回数を設定し、 k に 1 を代入する。そして全ての $M_{c_i \rightarrow v_{j''}}(x_{j''})$ ($1 \leq i \leq m, j'' \in \mathcal{J}(i), x_{j''} \in \{0, 1\}$) に 1 を代入する。ただし、 m はパリティ検査行列の行数を表す。

手順 2 変数ノード処理

全ての変数ノード v_j ($1 \leq j \leq n$) において、変数ノード処理 (式 (12)) を実行する。

手順 3 検査ノード処理

全ての検査ノード c_i ($1 \leq i \leq m$) において、検査ノード処理 (式 (13)) を実行する。

手順 4 パリティ検査処理

全ての変数ノード v_j ($1 \leq j \leq n$) において、一時推定処理 (式 (14)) を実行する。そして全てのシンボル x_j ($1 \leq j \leq n$) に対して一時推定シンボル

$$\tilde{x}_j = \begin{cases} 0 & g(x_j = 0) \geq g(x_j = 1), \\ 1 & \text{otherwise,} \end{cases}$$

を得て、一時推定語 $\tilde{\mathbf{x}} = \tilde{x}_1\tilde{x}_2\cdots\tilde{x}_n$ を得る。ここで、 $H\tilde{\mathbf{x}}^T = \mathbf{0}$ であれば $\tilde{\mathbf{x}}$ を推定語 $\hat{\mathbf{x}}$ として出力し終了する。それ以外の場合は手順 5 へ。

手順 5 終了判定

$k = K$ の場合は $\tilde{\mathbf{x}}$ を推定語 $\hat{\mathbf{x}}$ として出力し終了する。その他の場合には $k \leftarrow k + 1$ を代入し、手順 2 へ。

(注6)：誌面の都合により省略するが、読者自身の手でぜひ確かめて頂きたい。

ここまで、2部グラフが木の場合に限って説明してきたが、実用化されている誤り訂正符号のパリティ検査行列の2部グラフは木とはなっていない。2部グラフが木でない場合、sum-product アルゴリズムで計算した値は事後確率の近似値となるが、任意のノードに対して深さ6から8程度までの部分グラフ（以下、近傍グラフと呼ぶ）が木になっていれば、良好な復号特性を得られることが経験的に知られている。なお深さとは、任意のノード a をルートにして2部グラフを木のように展開したとき、経路に含まれるエッジの数のことを指す。

では、近傍グラフがなるべく木になるようにパリティ検査行列を構成するには、どうすればよいのだろうか？ その答えが実は LDPC 符号のパリティ検査行列を特徴づける、“疎”の部分にあることを次節で紹介する。

3.5 LDPC 符号のパリティ検査行列

LDPC 符号は「疎（低密度）なパリティ検査行列で定義される符号」と定義される。行列の非零要素^(注7)の割合を行列の密度として定義すると、疎なパリティ検査行列とは、行列の密度が低い（つまりパリティ検査行列中に含まれる非零要素数が少ない）パリティ検査行列を指す。1と0の出現確率が等しくなるように各要素を決めたランダム行列の密度が1/2となるのに対し、LDPC 符号のパリティ検査行列の密度は1/nに比例する程度である。つまり n が十分に大きいときには LDPC 符号のパリティ検査行列の密度は0に漸近するほど低密度となる。パリティ検査行列が疎であると、対応する2部グラフに含まれるエッジの数が少なくなるため、閉路が生成されにくくなる。つまり LDPC 符号のパリティ検査行列は、疎にすることで近傍グラフが木になりやすくなり、sum-product アルゴリズムで得られる事後確率の近似精度を上げることに成功しているのである。

以下では Gallager が彼の博士論文⁽³⁾で提案したパリティ検査行列の構成法を紹介する。この構成法は、1列に含まれる非零要素の数が l で、1行に含まれる非零要素の数が r となるような長さ n のパリティ検査行列 $H^{(n,l,r)}$ を生成する。ここでは説明を簡単にするため、 n は r の倍数であるものとする。

まず、サブ行列 $H_0^{(n,l,r)}$ を次のように定義する。

$$H_0^{(n,l,r)} := \begin{bmatrix} \mathbf{h}^{(n,r)} \\ s^{(r)}(\mathbf{h}^{(n,r)}) \\ s^{(2r)}(\mathbf{h}^{(n,r)}) \\ \vdots \\ s^{(n-r)}(\mathbf{h}^{(n,r)}) \end{bmatrix}$$

ただし $\mathbf{h}^{(n,r)}$ は先頭に非零要素が r 個連続で並び、残りは全て0となる長さ n のベクトルである。また $s^{(r)}(\mathbf{h})$ は、ベクトル $\mathbf{h}$ を右方向に r 要素分巡回シフトしたベクトルを返す関数とする。

このサブ行列 $H_0^{(n,l,r)}$ を使って、パリティ検査行列 $H^{(n,l,r)}$ が次のように得られる。

$$H^{(n,l,r)} = \begin{bmatrix} H_0^{(n,l,r)} \\ \pi_1(H_0^{(n,l,r)}) \\ \vdots \\ \pi_{l-1}(H_0^{(n,l,r)}) \end{bmatrix} \quad (26)$$

なお、 $\pi_i(H_0^{(n,l,r)})$ ($1 \leq i \leq l-1$) は、サブ行列 $H_0^{(n,l,r)}$ の列ベクトルをランダムに並べ替える関数とする。

例として、上記構成法で生成した $H^{(8,2,4)}$ を以下に示す。

$$H^{(8,2,4)} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Gallager の通信路符号化定理の証明

本稿の内容から逸れるが、Gallager の論文の中でおそらく LDPC 符号よりも有名な通信路符号化定理の証明^{(7)(注)}に関するエピソードを、コーヒーブレイクを兼ねてここで紹介する。

MIT の授業で彼が通信路符号化定理を教えることになったときのことである。彼は最初、当時出版されたばかりの Robert Fano によって書かれた情報理論の教科書を使って通信路符号化定理を説明するつもりだった。しかしその教科書に書かれた証明が複雑で、Gallager はよい説明方法を見つけることができなかった。授業でどのように通信路符号化定理を説明するか、4、5日眠れないほど悩んだ結果、非常にシンプルな証明が突然 Gallager の頭の中に浮かび上がってきたそうである。この証明方法こそが通信路符号化定理の証明に関する論文⁽⁷⁾で示された方法である。この証明は教育と研究がうまく調和したよい例であったと Gallager は述べている⁽⁹⁾。教えるために物事を十分に理解しようと試みることで、このほかにも彼は新しく素晴らしい研究アイデアを何度も得ることができたそうである。

(注)：本誌の解説論文⁽⁸⁾においても、この論文によって Gallager が情報理論における中興の祖と呼ばれるようになったと讃えている。

4. 実用的な LDPC 符号の設計

3.5 節では、Gallager の博士論文⁽³⁾で述べられている LDPC 符号の構成法を説明した。この構成法は簡易ながら十分な訂正能力をもつ LDPC 符号を設計することができる優れた構成法だが、符号長の長い LDPC 符号を構成する際には必ずしも実用的とはいえない。

LDPC 符号の復号アルゴリズムである sum-product アルゴリズムでは、3.3 節で述べたように変数ノード、検査ノードが計算する処理はインデックスによらず式 (12)、(13) でそれぞれ示した同一の計算処理を行う。そのため LDPC 符号の復号器をデジタル回路で構成する際、変数ノード、検査ノードの処理回路はそれぞれ符号長 n 、パリティ検査式数 m よりも少ない数

(注7)：2元の場合には1。

の回路を用意し、異なるインデックスの処理にも同一の回路を使いまわすことで、回路規模を抑えることができる。極端な例でいえば、一つの変数ノード処理回路、検査ノード処理回路をそれぞれ符号長 n 回、パリティ検査式数 m 回繰り返し用いれば、sum-product アルゴリズム繰り返し 1 回分の処理を実施することも可能である。このときデジタル回路としてはそれぞれ一つ分の処理回路だけ備えていればよいので、とても小さな回路で実現できる。

このように sum-product アルゴリズムでは変数ノード処理、検査ノード処理の計算自体は共通化できるものの、各ノードで計算に用いる値のインデックス集合 $\mathcal{I}(j)$, $\mathcal{J}(i)$ が異なる。

例えば Gallager の構成法の場合、 $\frac{n}{r} + 1$ 行め以降のパリティ検査式では、ランダム置換により各行のインデックス集合 $\mathcal{J}(i)$ に規則性はない。そのため検査ノードの処理回路 i が $1 \leq j \leq n$ の全ての変数ノードに接続できるよう、選択対象数が大きな選択回路を用意しなければならないが、選択対象数が増えるほど選択回路の複雑度が増加するため、大きな回路になってしまう。

そこで実用的には、一定数の変数ノードと一定数の検査ノードの接続に規則性をもたせ、接続構造を限定した 2 部グラフで定義された LDPC 符号が用いられる。その代表的な LDPC 符号である原始グラフに基づく準巡回 LDPC 符号 (Quasi-Cyclic Protograph-based LDPC Code) について次節以降で紹介していく。

4.1 原始グラフに基づく準巡回 LDPC 符号

原始グラフに基づく LDPC 符号とは、符号を定義する 2 部グラフが、比較的小さなグラフ (原始グラフ: *protograph*) の拡大表現により与えられる LDPC 符号と定義される⁽¹⁰⁾。原始グラフはノード間の接続関係を表現する $m_p \times n_p$ の基本行列 $\mathbf{B}$ で記述される。基本行列 $\mathbf{B}$ の各要素 $\mathbf{B}(i, j) \in \mathbb{Z}^+$, $i \in \{1, \dots, m_p\}$, $j \in \{1, \dots, n_p\}$ はノード間のエッジ数を表す。原始グラフに基づく LDPC 符号のパリティ検査行列は、複製置換 (*copy-and-permute*) またはリフティング (*lifting*)⁽¹⁰⁾ と呼ばれるグラフ拡大操作によって、ベースとなる基本行列から生成される。

1×2 の基本行列で定義される原始グラフから、複製置換によって 4×8 のパリティ検査行列 (2 部グラフ) を構成する様子を図 5 を用いて説明する。図 5 中、最も左にある変数ノードが 2 個、検査ノードが 1 個のグラフが原始グラフである。この原始グラフは次の二つの特徴をもつ。

- (1) 変数ノードの半数 (2 個中 1 個) が次数 2 の変数ノードで、残りの半数が次数 1 の変数ノードである。なお次数とはノードに接続しているエッジの数を表す。
- (2) 検査ノードは全て (といっても 1 個だが) エッジを 3 本もち (次数 3)、そのうち 2 本のエッジを次数 2 の変数ノードに接続し、1 本を次数 1 の変数ノードに接続している。

図 5 中では、複製置換後も元の変数ノードが分かりやすいよう次数 2 の変数ノード及びノードに接続しているエッジを赤色、次数 1 の変数ノード及びノードに接続しているエッジを青色に

図 5 1×2 の基本行列で定義される原始グラフから、複製置換によって 4×8 のパリティ検査行列 (2 部グラフ) を構成する様子

している。

複製置換は複数ステップで実施される。図 5 では 2 ステップの複製置換が実施されている。最初のステップでは、基本行列 $\mathbf{B}$ 中の要素 $\mathbf{B}(i, j)$ で最大の数 $b_{\max} = \max(\mathbf{B}(i, j))$ の大きさの正方行列を基本行列中の要素へ重畳する。このとき、重畳する正方行列の各行、各列の重み (1 の数) は $\mathbf{B}(i, j)$ となるようにする。図 5 では $\mathbf{B}(1, 1) = 2$, $\mathbf{B}(1, 2) = 1$ なので、 $\mathbf{B}(1, 1)$ には

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix},$$

$\mathbf{B}(1, 2)$ には

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

を重畳し、要素として 0, 1 のみをもつ $\tilde{\mathbf{B}}$ (図 5 中央) を得る。元の原始グラフ (図 5 左) では 2 本のエッジ (赤) が同一の変数ノードと検査ノードにつながっていた。このように同一のノード間に接続するエッジをパラレルエッジとも呼ぶ。複製置換の最初のステップの目的は、このようなパラレルエッジを除去することである。

次のステップでは、 $\tilde{\mathbf{B}}$ の非零要素に置換行列

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \text{ 若しくは } \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

を重畳し、更に零要素には全ての要素が零の正方行列

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

を重畳し、所望のパリティ検査行列 $\mathbf{H}$ を得る。この操作をグラフの操作として見ると、もとの原始グラフを複製し、その後エッジを置換することによって大きな 2 部グラフを生成する操作に対応することから、複製置換と呼ばれる。

複製置換による 2 部グラフ生成のよいところは、生成された 2 部グラフが原始グラフが備えていたグラフの構造 (図 5 でいえば、先に述べた 2 つの特徴) を保持できる点である。実際に図 5 の最右の 2 部グラフを見ると、原始グラフが備えていた 2 つの特徴を保持し続けていることが分かる。

LDPC 符号の訂正能力は符号を定義する 2 部グラフ (パリティ

ティ検査行列)の構造によって決定される。そこで LDPC 符号を設計する際には符号に要求される制約条件を満たしつつ最良の訂正能力を持つ 2 部グラフを構成することになるが、符号長が長くなると設計自由度が非常に大きくなるため、2 部グラフを直接設計するのは困難になる。

そこで比較的小さいグラフである原始グラフでグラフ構造の最適化を行い、その構造を保持した 2 部グラフを複製置換により生成するというのが、実用的な符号設計手法としてよく用いられている^{(11), (12)}。次節ではこの原始グラフの設計方法について述べる。

4.2 原始グラフの設計

一般に LDPC 符号に求められる特性は

- (1) 良好な復号しきい値
- (2) 線形最小距離増加性

の二つである。(1)は符号長及び復号の繰り返し回数を十分大きくしたときに復号誤り率を任意に小さくできる通信路誤り率の最大値を意味し、復号しきい値が高いほど、復号誤り率が 1 付近から急しゅんに減少する“ウォーターフォール領域”が通信路誤り率が大きいときに始まるようになる。また (2)は符号の最小距離が符号長に比例して大きくなる特性⁽¹³⁾の有無に対応するもので、LDPC 符号の復号誤り率カーブにおいて復号誤り率の減少が、ウォーターフォール領域と比べゆるやかな“エラーフロア領域”の性能に関連する。具体的には最小距離が大きいほど、復号誤り率が小さくなるまでエラーフロア領域が現れにくくなる。原始グラフの設計においては、(2)を保持しながら (1)をできるだけ大きくする原始グラフを設計することが目標となる。

復号しきい値は Richardson と Urbanke によって開発された密度発展法⁽⁶⁾により求める。ただし、原始グラフに基づく LDPC 符号は構造をもつため、マルチエッジ型の密度発展法により計算される⁽¹⁴⁾。密度発展法による厳密計算は計算量が大きいため、原始グラフの復号しきい値の計算には EXIT 関数を使った PEXIT と呼ばれる近似計算が用いられることが多い⁽¹¹⁾。

原始グラフの構造を設計することはつまり等価表現である基本行列 $\mathbf{B}$ の各要素を決定することに等しく、 $m_p \times n_p$ の基本行列 $\mathbf{B}$ の探索範囲はただか $(1 + b_{\max})^{m_p n_p} - 1$ である。ただし、 $b_{\max}$ は基本行列 $\mathbf{B}$ の要素の値の最大値である。基本行列 $\mathbf{B}$ のサイズが大きいほど復号しきい値の高い LDPC 符号が得られることが知られている⁽¹⁵⁾が、基本行列のサイズに応じて探索範囲が指数的に大きくなるため、網羅的に探索することが困難になる。

以上をまとめると、原始グラフの設計作業とは、生成される符号が線形最小距離増加性を保持するという制約条件のもと、復号しきい値を最大化させるという離散最適化問題と考えることができ。探索範囲が大きな場合には焼きなまし法や差分進化法などのヒューリスティックな最適化アルゴリズムにより解を導出する^{(10), (16)}。

以下では文献(16)で提案された差分進化法による基本行列の設計手順を説明する。差分進化法は進化系パラメータ最適化ア

ルゴリズムの一つで、任意のコスト関数に従い候補解の改善操作を繰り返し試行して最適解を見つける。基本行列の設計においては、復号しきい値をコスト関数として候補解を改善していく。以下に具体的なアルゴリズムを記述する。

差分進化法による基本行列の設計アルゴリズム

(準備) 初期基本行列 $\mathbf{B}_{init}$ 、構造を固定する列インデックスの集合 F 、交叉確率 p_c 、要素の最大数 $b_{\max}$ 、候補となる基本行列の数 S 、世代数 G 、そして基本行列が満たすべき制約条件 Γ を定める。

手順 1 (初期化) まず、 $\mathbf{B}_1^{(0)}$ に $\mathbf{B}_{init}$ をセットする。そして $\mathbf{B}_1^{(0)}$ のうち、 $j \in F$ 以外の全ての列の各要素について、 $\{0, 1, \dots, b_{\max}\}$ の中からランダムに選んだ値で置換した $S - 1$ 個の基本行列 $\mathbf{B}_2^{(0)}, \mathbf{B}_3^{(0)}, \dots, \mathbf{B}_S^{(0)}$ を生成する。このとき、 $\mathbf{B}_s^{(0)}, 2 \leq s \leq S$ はあらかじめ定めた条件 Γ を満たすように生成する。

手順 2 (変異) 世代 g の変異行列 $\mathbf{M}_s^{(g)}, 1 \leq s \leq S$ を次式で求める。

$$\mathbf{M}_s^{(g)} = \Phi(\mathbf{B}_{r_1}^{(g)} + 0.5(\mathbf{B}_{r_2}^{(g)} - \mathbf{B}_{r_3}^{(g)})),$$

ただし r_1, r_2, r_3 は $[1, S]$ の範囲からランダムに選んだ整数で、 $\Phi(\mathbf{X})$ は行列 $\mathbf{X}$ の各要素の値を最も近い整数に丸めた要素をもつ行列を返す関数である。

手順 3 (交叉) 次世代の候補解 $\mathbf{N}_s^{(g)}, 1 \leq s \leq S$ を次式で求める。

$$\mathbf{N}_s^{(g)}(i, j) = \begin{cases} \mathbf{M}_s^{(g)}(i, j) & \text{with probability } p_c, \\ \mathbf{B}_s^{(g)}(i, j) & \text{otherwise,} \end{cases}$$

ただし $1 \leq i \leq m_p, 1 \leq j \leq n_p$ である。

手順 4 (選択) 世代 $g + 1$ の解の候補群 $\mathbf{B}_s^{(g+1)}, 1 \leq s \leq S$ を次のように選択する。

$$\mathbf{B}_s^{(g+1)} = \begin{cases} \mathbf{N}_s^{(g)} & \text{if } \delta(\mathbf{N}_s^{(g)})\mathbb{I}(\mathbf{N}_s^{(g)}) > \delta(\mathbf{B}_s^{(g)}), \\ \mathbf{B}_s^{(g)} & \text{otherwise,} \end{cases}$$

ただし $\delta(\mathbf{X})$ は $\mathbf{X}$ の構造をもつ原始グラフに基づく LDPC 符号アンサンブルの復号しきい値で、 $\mathbb{I}(\mathbf{X})$ は $\mathbf{X}$ が条件 Γ を満たすときに 1、そうでないときは 0 を返す指示関数である。

手順 5 (終了) 手順 2 から手順 4 を世代 G になるまで繰り返し行い、最も高い復号しきい値を与えた基本行列 $\mathbf{B}_s^{(g)}$ を解として出力する。

上記のアルゴリズムを用いて符号化率 0.5 の基本行列を設計してみよう。今回は $m_p = 4, n_p = 8$ の基本行列を設計する。一般に列の要素の合計が 2 となる列 (重み 2 の変数ノードに対応) を含む原始グラフから生成される LDPC 符号の復号しきい値は高くなるため⁽¹⁵⁾、初期行列 $\mathbf{B}_{init}$ として以下を与えた。

$$\mathbf{B}_{init} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix},$$

次数 2 の変数ノードを保持したいため、固定する列インデックスの集合を $F = \{8\}$ とする。基本行列中の要素が大きすぎると、その部分のエッジが密になりショートサイクルを生じやすくなるため、要素の最大値は $b_{\max} = 3$ とした。そして交叉確率 p_c は $0.88^{(17)}$ 、 $S = 5000$ 、 $G = 5000$ とした。

そして制約条件 Γ は

- (1) 基本行列中の要素は 0 以上 $b_{\max}$ 以下。
- (2) 各行の要素の和（検査ノード次数に対応）は 2 以上。
- (3) 各列の要素の和（変数ノード次数に対応）は 3 以上^(注8)。
- (4) どの列も要素の和（変数ノード次数の最大値）が 8 以下^(注9)。

とした。

上述したパラメータを用いて差分進化法を実行し、候補となる基本行列 $\hat{\mathbf{B}}$ を得た。このようにして得られた基本行列を微調整すると更により基本行列が見つかる可能性が高いため、部分的にブルートフォース探索をすることで復号しきい値がより高い基本行列を探索した。部分的なブルートフォース探索とは、 F のものを除いた $\hat{\mathbf{B}}$ の三つの列をランダムに選択し、これらの列の要素に対してブルートフォース探索を実行するというものである。このような微調整は基本行列が変化しなくなるまで繰り返す。注意点として、部分的なブルートフォース探索は、局所的な最適解に近いものを求めるだけなので、差分進化法を用いずにこの探索だけでは最適な基本行列を見つけることが難しいことはここで述べておく。

このように差分進化法を実行した結果、基本行列

$$\mathbf{B}_{1/2} = \begin{bmatrix} 3 & 3 & 0 & 0 & 1 & 0 & 0 & 0 \\ 2 & 3 & 0 & 1 & 0 & 1 & 0 & 0 \\ 3 & 2 & 1 & 2 & 0 & 0 & 1 & 1 \\ 0 & 0 & 2 & 0 & 2 & 2 & 2 & 1 \end{bmatrix}. \quad (27)$$

を得た。加法的白色ガウス通信路^(注10)における基本行列 $\mathbf{B}_{1/2}$ の復号しきい値は 0.501 dB ^(注11)で、NASA の深宇宙通信規格に採用された AR4JA 符号⁽¹³⁾の 0.628 dB よりもシャノン限界に近い。ちなみに符号化率 $\frac{1}{2}$ のシャノン限界は 0.187 dB である。

(注8)：これは線形最小距離増加性をもつために必要な条件。

(注9)：探索範囲が限定されるとともに複製置換後の 2 部グラフに短いサイクルが含まれないようにするため。

(注10)：これまで本稿では加法的白色ガウス通信路についての説明を与えていないが、文献(13)に記載の復号しきい値と比較するためにここではこの通信路での復号しきい値を示していることをご容赦願いたい。

(注11)：デシベルの復号しきい値は $10 \log_{10} \frac{1}{2v_{th}R}$ で計算される。ただし v_{th} は PEXIT で計算される復号可能な最大の雑音電力（分散）で、 R は符号化率である。

4.3 巡回置換行列を用いた複製置換

前節で得られた基本行列から、符号を定義するパリティ検査行列を生成する方法について述べる。原始グラフに基づく LDPC 符号では、基本行列からパリティ検査行列を生成する際には 4.1 節で述べたように複製置換を行う。このときの置換行列は、ランダム置換行列ではなく回路実装が容易な巡回置換行列を用いるのが一般的である。大きさが $q \times q$ の巡回置換行列を次のように定義する。

$$\mathbf{P}(\sigma) = \begin{bmatrix} s^{(\sigma)}(\mathbf{h}^{(q,1)}) \\ s^{(1)}(s^{(\sigma)}(\mathbf{h}^{(q,1)})) \\ \vdots \\ s^{(q-1)}(s^{(\sigma)}(\mathbf{h}^{(q,1)})) \end{bmatrix} \quad (28)$$

ただし σ は巡回シフト値で $0, 1, \dots, q-1$ の値をとる。長さ q のベクトル $\mathbf{h}^{(q,1)}$ と巡回シフト関数 $s^{(\sigma)}$ はそれぞれ 3.5 節で定義したものである。式 (28) の定義より、 $\mathbf{P}(0)$ はサイズ q の単位行列となる。このように巡回置換行列を用いて得られる LDPC 符号は準巡回 LDPC 符号とも呼ばれる⁽¹⁸⁾。

4.1 節では 2 ステップの複製置換を述べたが、3 ステップ以上の複製置換を実施してもよい。 $\tilde{m}_p$ を最終ステップ前の基本行列 $\tilde{\mathbf{B}}$ の行数としたとき、準巡回 LDPC 符号の最小距離の上界は $(\tilde{m}_p + 1)! - 2(\tilde{m}_p - 2)!$ となるため⁽¹⁹⁾、2 ステップで大きなサイズの巡回置換行列を用いて所望の大きさのパリティ検査行列を生成するよりも、複製置換を複数ステップ行い、最終ステップ前の基本行列 $\tilde{\mathbf{B}}$ をある程度大きくしておいたほうが、誤り訂正能力の観点からは好ましい。

一方で回路実装の観点からは、巡回置換行列のサイズは復号処理の回路上的で並列処理単位となるので、訂正処理のスピードと回路規模のトレードオフから適切に設定することが望ましい。

このように誤り訂正能力や回路実装の観点から巡回置換行列のサイズを決め、最終ステップで基本行列 $\tilde{\mathbf{B}}$ 中の非零要素へ巡回置換行列を重畳していく。このとき、各巡回置換行列の巡回シフト値の決め方によっては、3.3、3.4 節で述べた短い閉路が生まれてしまい、LDPC 符号の訂正能力が劣化してしまうことがあるので、巡回シフト値は注意深く決めなければならない。

巡回シフト値を決めるアルゴリズムとしては、閉路の最小の長さ（内径：*girth*）が小さくならないように貪欲法を用いて決定していく Progressive edge growth (PEG) アルゴリズム⁽²⁰⁾がよく知られている。

PEG は内径が小さくならないように 2 部グラフのエッジの接続先を一つずつ決めていくアルゴリズムであるため、通常は各エッジの接続先を追跡して閉路の長さを一つ一つ計測する必要があるが、巡回置換行列を用いた準巡回 LDPC 符号では、巡回置換行列の巡回シフト値を使って閉路の判定を行うことができる。

基本行列 $\tilde{\mathbf{B}}$ 中の非零要素 $\tilde{\mathbf{B}}(i, j)$ へ重畳する巡回置換行列の巡回シフト値を $\sigma_{(i, j)}$ とすると、例えば、 $\tilde{\mathbf{B}}(i_1, j_1)$ 、 $\tilde{\mathbf{B}}(i_1, j_2)$ 、 $\tilde{\mathbf{B}}(i_2, j_2)$ 、 $\dots$ 、 $\tilde{\mathbf{B}}(i_n, j_n)$ 、 $\tilde{\mathbf{B}}(i_n, j_1)$ に重畳する巡回置換行列

図 6 長さ 4 の閉路を構成する四つの巡回シフト値の組と対応するパリティ検査行列

の組により長さ $2n$ の閉路を構成する場合、これらの巡回置換行列の巡回シフト値の間には次の関係が成り立つ⁽¹⁸⁾。

$$\begin{aligned} & \sigma(i_1, j_1) - \sigma(i_1, j_2) + \sigma(i_2, j_2) - \dots \\ & + \sigma(i_n, j_n) - \sigma(i_n, j_1) = 0 \pmod{q}, \end{aligned} \quad (29)$$

ただし mod は剰余演算を表す。

図 6 に長さ 4 の閉路になる巡回置換行列の組の例を示す。図 6 中の左の行列は、 $\sigma(i, j)$ を要素にもつ行列である。図 6 中の右の行列は、サイズ 5 ($q = 5$) の巡回置換行列を重畳したパリティ検査行列である。図 6 中の左に示すこの四つの巡回シフト値の組は、 $1 - 2 + 3 - 2 = 0$ となるので、式 (29) の関係を満たし、長さ 4 の閉路を構成する。実際に対応する図 6 中の右のパリティ検査行列の赤い実線の矢印で結ばれた部分を見ると、パリティ検査行列が長さ 4 の閉路を構成していることが分かる。更に図 6 中の右のパリティ検査行列をよく見てみると、長さ 4 の閉路を構成するエッジは赤い実線矢印だけでなく、ほかにも破線矢印で示した閉路が存在し、合わせて巡回置換行列のサイズの数だけ長さ 4 の閉路が存在していることが分かる。

実は巡回置換行列で構成されたパリティ検査行列の内径を、12 より大きな値にすることは難しい。なぜならば、巡回置換行列で生成されたパリティ検査行列が 2×3 の巡回シフト値の組をもつ場合、必ず長さ 12 の閉路をもってしまう。これは図 7 に示すような閉路を考えることで、式 (29) の関係式を使って以下のように容易に示すことができる。

$$\begin{aligned} & \sigma(j, k+2) - \sigma(j+1, k+2) + \sigma(j+1, k+1) - \sigma(j, k+1) \\ & + \sigma(j, k) - \sigma(j+1, k) + \sigma(j+1, k+2) - \sigma(j, k+2) \\ & + \sigma(j, k+1) - \sigma(j+1, k+1) + \sigma(j+1, k) - \sigma(j, k) \\ = & (\sigma(j, k) - \sigma(j, k)) + (\sigma(j, k+1) - \sigma(j, k+1)) \\ & + (\sigma(j, k+2) - \sigma(j, k+2)) + (\sigma(j+1, k) - \sigma(j+1, k)) \\ & + (\sigma(j+1, k+2) - \sigma(j+1, k+2)) + (\sigma(j+1, k+1) - \sigma(j+1, k+1)) \\ = & 0 \end{aligned}$$

巡回シフト値の決定方法に関しては閉路の長さだけでなく、その閉路に含まれる変数ノードと、閉路外の検査ノードとの接続度合いを近似的に表す Approximate Cycle Extrinsic message degree (ACE)⁽²¹⁾ を用いたアルゴリズムも提案されている。長さ $2n$ の閉路の ACE は、 $\sum_{l=1}^n (d_l - 2)$ により計算される。ただし、 d_l は閉路に含まれる l 番目の変数ノードの次数である。

図 7 2×3 の巡回シフト値の組が長さ 12 の閉路を構成することを示す例

ACE とは閉路に含まれる変数ノードが閉路の外から受ける影響の量を近似的に表しており、ACE の値が大きいくほど、その閉路を含むトラッピングセット^(注12)のサイズが大きくなることが知られている⁽²¹⁾。

巡回シフト値を決定するアルゴリズムについては、本節で述べた以外にも数多くのアルゴリズムが提案されているので、アルゴリズムに興味のある読者はぜひ調べて頂きたい。

5. 2 元対称通信路での復号誤り率特性

最後に、2 元対称通信路における LDPC 符号の復号誤り率に関する数値実験結果を図 8 に示す。評価した LDPC 符号は、3.5 節の式 (26) の構成法で生成した列と行の非零要素数がそれぞれ $l = 3$, $r = 6$ である 510×1020 のパリティ検査行列で定義される LDPC 符号 (Gallager LDPC) と、4.2 節の式 (27) の基本行列から 4.3 節で説明した巡回置換行列を用いた複製置換により生成した 512×1024 のパリティ検査行列で定義される準巡回 LDPC 符号 (QC P-LDPC) である。復号処理は最大繰返し回数を 100 回とした同時更新型 sum-product 復号を用いた。また比較のため、ランダム誤りに対する誤り訂正符号として一般に用いられる BCH (Bose-Chaudhuri-Hocquenghem) 符号⁽²³⁾ を限界距離復号した際の結果も図 8 に示す。BCH 符号は長さが 1023 ビットで設計距離 103 (訂正可能ビット数 51 ビット) の符号を用い、符号長及び伝送速度が評価する LDPC 符号と同等になるようにした。

図 8 を見ると、通信路誤り率が 0.06 のとき、BCH 符号の復号誤り率はほぼ 1 になってしまうのに対し、LDPC 符号の復号誤り率は Gallager LDPC が $\frac{1}{100}$ 、QC P-LDPC が $\frac{1}{200}$ を達成していることが分かる。

6. む す び

本稿では、確率推論の観点から最適な最大事後確率復号が、分配則を用いることで効率良く実行できることを紹介し、具体的な復号アルゴリズムである sum-product 復号を与えた。更に sum-product 復号によって精度の高い最大事後確率が計算できるパリティ検査行列の構成として、LDPC 符号のような疎な構成が適していることを述べた。

次に実用的な LDPC 符号である原始グラフに基づく準巡回

(注12) : sum-product 復号が復号失敗となる誤りビットの集合⁽²²⁾。sum-product 復号を行う LDPC 符号では、最小距離よりもトラッピングセットの最小サイズの方が復号特性 (特にエラーフロア領域) に大きな影響を与える。

図8 2元対称通信路における復号誤り率に関する数値実験結果

LDPC 符号を設計する手法について説明した。LDPC 符号の設計手法に関して日本語で解説した文献はあまりないので、本稿が設計手法理解の一助になれば幸いである。

本稿はなるべく直感的な理解が得られるよう、厳密な定理や証明を与えるよりは、例を通してそのエッセンスを紹介した。もっと LDPC 符号について知りたい方で、その深い数理に興味がある方には文献(24)を参照することをおすすめする。一方、本稿で説明した以外の LDPC 符号の設計手法や本稿では紹介できなかった各種復号アルゴリズムなど、より実用的なトピックに興味がある方には文献(25)が参考になるであろう。LDPC 符号の発展的なトピックである非 2 元 LDPC 符号や空間結合 LDPC 符号などについて、本学会の情報理論研究会の予稿に素晴らしい解説論文(26)・(27)があるので、ぜひご参照願いたい。

最後に、30 年以上という長い年月を経てその発明が評価されることとなった Gallager が、2020 年(第 36 回)日本国際賞の受賞者発表記者会見で研究者へ宛てたメッセージをもって、筆を置こうと思う。

Don't necessarily be upset at the idea that what you do is not useful, because perhaps it will be useful later. Do something which is novel, which is interesting, and what you hope will be useful in the future.

(筆者訳：あなたの研究が役に立たないと思ったり、思われたりしても動揺する必要はありません。しばらくしてから役に立つかもしれないからです。将来役に立つことを期待して、新しいこと、面白いことを研究しましょう。)

文 献

(1) 萩原学(編著), 進化する符号理論, 日本評論社, 2016.

(2) C.E. Shannon, "A mathematical theory of communication," Bell System Technical Journal, vol. 27, no. 3, pp. 379–423, 1948.

(3) R.G. Gallager, Low-Density Parity-Check Codes, MIT Press, 1963.

(4) D.J.C. MacKay, "Good error-correcting codes based on very sparse matrices," IEEE Trans. Inf. Theory, vol. 45, no. 2, pp. 399–431, March 1999.

(5) T.J. Richardson and R. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," IEEE Trans. Inf. Theory, vol. 47, no. 2, pp. 599–618, 2001.

(6) T.J. Richardson, M. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," IEEE Trans. Inf. Theory, vol. 47, no. 2, pp. 619–637, 2001.

(7) R. Gallager, "A simple derivation of the coding theorem and some applications," IEEE Trans. Inf. Theory, vol. 11, no. 1, pp. 3–18, Jan. 1965.

(8) 笠原正雄, "リード・ソロモン符号の半世紀," 信学 FR 誌, vol. 5, no. 1, pp. 28–41, 2011.

(9) "Robert G. Gallager Wins the 1999 Harvey Prize." https://wayback.archive-it.org/all/20070417175505/http://www.ee.ucla.edu/~congshen/robert_gallager.pdf

(10) J. Thorpe, "Low density parity check (LDPC) codes constructed from protographs," JPL IPN Progress Report, 42-154, pp. 1–7, Aug. 2003.

(11) G. Liva and M. Chiani, "Protograph LDPC codes design based on EXIT analysis," IEEE Global Telecommunications Conference (GLOBECOM), pp. 3250–3254, Nov. 2007.

(12) Y. Fang, G. Bi, Y.L. Guan, and F.C.M. Lau, "A survey on protograph LDPC codes and their applications," IEEE Commun. Surveys Tuts., vol. 17, no. 4, pp. 1989–2016, 2015.

(13) D. Divsalar, S. Dolinar, C.R. Jones, and K. Andrews, "Capacity-approaching protograph codes," IEEE J. Sel. Areas Commun., vol. 27, no. 6, pp. 876–888, 2009.

(14) T.J. Richardson and R. Urbanke, "Multi-edge type LDPC codes," 2004. <https://www.researchgate.net/publication/37439748.Multi-edge-type-LDPC-codes>

(15) T.V. Nguyen, A. Nosratinia, and D. Divsalar, "The design of rate-compatible protograph LDPC codes," IEEE Trans. Commun., vol. 60, no. 10, pp. 2841–2850, 2012.

(16) H. Uchikawa, "Design of non-precoded protograph-based LDPC codes," IEEE International Symposium on Information Theory (ISIT), pp. 2779–2783, June 2014.

(17) A.K. Pradhan, A. Subramanian, and A. Thangaraj, "Deterministic constructions for large girth protograph LDPC codes," IEEE International Symposium on Information Theory (ISIT), pp. 1680–1684, July 2013.

(18) M.P.C. Fossorier, "Quasi-cyclic low-density parity-check codes from circulant permutation matrices," IEEE Trans. Inf. Theory, vol. 50, no. 8, pp. 1788–1793, 2004.

(19) R. Smarandache and P.O. Vontobel, "Quasi-cyclic LDPC codes: Influence of proto- and Tanner-graph structure on minimum hamming distance upper bounds," IEEE Trans. Inf. Theory, vol. 58, no. 2, pp. 585–607, Feb. 2012.

(20) D.M. Arnold, E. Eleftheriou, and X.Y. Hu, "Progressive edge-growth Tanner graphs," IEEE Global Telecommunications Conference (GLOBECOM), vol. 2, pp. 995–1001, 2001.

(21) T. Tian, C. Jones, J. Villasenor, and R.D. Wesel, "Selective avoidance of cycles in irregular LDPC code construction," IEEE Trans. Commun., vol. 52, no. 8, pp. 1242–1247, 2004.

(22) T.J. Richardson, "Error floors of LDPC codes," 41st Annual Allerton Conference on Communications,

- pp. 1426–1435, 2003.
- (23) S. Lin and D.J. Costello, Error Control Coding, 2 edition, Prentice Hall, 2004.
 - (24) T.J. Richardson and R. Urbanke, Modern Coding Theory, Cambridge University Press, 2008.
 - (25) W.E. Ryan and S. Lin, Channel Codes: Classical and Modern, Cambridge University Press, 2009.
 - (26) 笠井健太, “多元 LDPC 符号とその応用,” 信学技報, vol. 110, no. 205, pp. 1–6, Sept. 2010.
 - (27) 笠井健太, “空間結合符号とその応用,” 信学技報, vol. 111, no. 220, pp. 1–8, Sept. 2011.

(IT 研究会提案, 2020 年 9 月 11 日受付,
2020 年 10 月 13 日再受付)

内川浩典 (正員: シニア会員)

2003 横浜国立大学大学院修了. 同年株式会社東芝入社. 2012 東京工業大学博士 (工学). 2013～2014 カリフォルニア大学サンディエゴ校客員研究員. 2019 キオクシア株式会社 (旧 東芝メモリ). フラッシュメモリ向け誤り訂正符号の研究開発に従事. IEEE シニア会員.