シフト演算を利用した噴水符号

Zigzag Decodable Fountain Codes

野崎隆之 Takayuki NOZAKI

アブストラクト 誤り訂正符号はデジタル情報に生じる誤りを訂正する基礎技術であり、信頼性が高い情報システムを実現する。消失訂正符号は誤り訂正符号の一種であり、デジタル情報の消失を訂正する技術である。噴水符号は消失訂正符号の一つであり、ネットワークにおける一対多同報通信であるマルチキャストの信頼性を向上させることができる。本稿では、消失訂正符号と既存の噴水符号を概説した後に、筆者が提案したシフト演算を利用した噴水符号について説明し、その性能を比較する。

キーワード 誤り訂正符号, 噴水符号, シフト演算

Abstract Error-correcting codes, which are fundamental methods for correcting errors occurring in digital information, realize reliable information systems. Erasure-correcting codes are a class of error-correcting codes and correct erasures in digital information. Fountain codes are erasure-correcting codes providing reliable multicasting, which sends the same message to multiple destinations. In this paper, we briefly review existing erasure-correcting and fountain codes. Then, we introduce the zigzag decodable fountain codes proposed by the author. Moreover, we compare their performance with an existing fountain code called Raptor code.

Key words Error-correcting code, Fountain code, Shift operation

1. はじめに

パケット通信とは、メッセージを複数のパケットに変換して、受信者に伝送する方式である。インターネットはパケット通信の代表的な一例である。パケット通信において、正確な情報伝達を達成するには、全てのパケットを正しく受信する必要がある。しかしながら、パケット通信においては、一部のパケットが受信者に届かず、消失する。したがって、受信者は何らかの方法で消失したパケットを得る必要がある。

送信者と受信者がそれぞれ一人づつの場合、すなわち1対1の通信の場合は、受信者は受信できなかったパケットを送信者に伝え、送信者が該当のパケットを送り直せばよい。このような手続きをパケットの再送と呼ぶ。一方、受信者が多数いる場合、すなわちマルチキャストの場合は、送信者は受信者からの再送の要求に応えることが困難になる。したがって、マルチキャストの場合は、送信の仕方を工夫し、再送せずに受信側で消失したパケットを復元する仕組みが必要になる。

消失訂正符号は符号化によって消失したデジタル情報を復元する技術である。送信者はメッセージを基に符号語と呼ばれ

る冗長性を有した系列を作成し、符号語を受信者に送る。通信の結果、受信者は符号語の一部が消失した系列を受け取る。受け取った系列を受信語という。符号語は冗長性を有しているため、受信者はこの冗長性を利用して消失したシンボルを復元することができる。この操作を復号という。

消失訂正符号の性能は符号化率(レート)と復号誤り率によって評価される。符号化率は符号語の長さでメッセージの長さの比で表され、情報伝達の効率を表現している。符号長が n でメッセージの長さが k の消失訂正符号の符号化率は k/n である。このように符号長が一定である符号をブロック符号という。一方、復号誤り率は復号において消失の訂正に失敗する確率である。性能の良いブロック消失訂正符号とは、符号化率が高く、復号誤り率の低いものを指す。性能の良いブロック消失訂正符号には、例えば低密度パリティ検査(LDPC: Low-Density Parity-Check)符号が挙げられる。ブロック消失訂正符号は、1対1通信のように、通信で発生する消失の割合(通信路消失率)が定まっているときに、適切な符号を設計することができる。言い換えると、通信路消失率に応じて、適切な符号化率は異なる。したがって、マルチキャストのように、受信者によって通信路消失率が異なる(注1)通信では、ブロック消失訂正符号を用いるのは適切ではない(注2)。

(注1): 受信者によって利用するネットワークの経路や環境が異なるため、通信路消失率は受信者に依存する。

(注2): 通信路消失率の低い受信者に合わせてブロック消失訂正符号を設計してしまうと、通信路消失率の高い受信者は復号が失敗してしまう。逆に、通信路消失率の高い受信者に合わせてブロック消失訂正符号を設計してしまうと、符号化率が低くなってしまい、通信路消失率が低い受信者も多く情報を受信する必要が生じる。このことについては、2. で詳しく説明する。

野崎隆之 正員 山口大学大学院創成科学研究科理学系学域

E-mail tnozaki@yamaguchi-u.ac.jp
Takayuki NOZAKI, Member (Graduate School of Sciences and Technology for Innovation, Yamaguchi University, Yamaguchi-shi, 753-8512, Japan).

電子情報通信学会 基礎・境界サイエティ
Fundamentals Review Vol.13 No.1 pp.7-19 2019 年 7 月
©電子情報通信学会 2019

図1 噴水符号を用いたマルチキャストの模式図 情報パケットと書かれた箇所の右側にある白丸は情報パケットを表しており、送信パケットと書かれた箇所の右側にある黒丸は送信パケットを表している。各受信者の右側に書かれた黒丸はその受信者が消失なく受けることのできた送信パケットを表している。

噴水符号⁽¹⁾はマルチキャストに適した消失訂正符号である。噴水符号においては、送信者は情報パケットを符号化し、十分な数の送信パケットを作成し、送信パケットを送る。受信者は送信パケットを一定の個数だけ受け取り、情報パケットを復号する。このとき、受信者が受け取る送信パケットの個数は情報パケットの個数よりも僅かに多い。余分に受け取る送信パケットの割合をパケットオーバーヘッドと呼び、噴水符号の性能指標に用いる。噴水符号を用いたマルチキャストにおいては、受信者が受け取った送信パケットを基に情報パケットを復号するため、受信者は送信者にパケットの再送を要求する必要がある。

図1は噴水符号を用いたマルチキャストの例の模式図を与えている。図中に示されている丸はパケットを表現している。情報パケットと書かれた箇所の右側にある白丸は情報パケットを表しており、送信パケットと書かれた箇所の右側にある黒丸は送信パケットを表している。各受信者の右側に書かれた黒丸はその受信者が消失なく受けることのできた送信パケットを表している。受信者Aのように、通信の状況が良く、パケットの消失がほとんど生じない受信者は、初めの方の送信パケットを受信することで、素早く情報パケットを復号することができる。受信者Bのように、通信の状況が悪く、パケットの消失が頻発するような受信者であっても、時間を掛けて一定数のパケットを受信できれば、元の情報パケットを復号することができる。したがって、噴水符号を用いれば、受信者の通信の状況に依存せず正確な情報伝達が可能になる。加えて、噴水符号を用いると、受信者Cのように、遅れて受信を開始した受信者であっても、一定数のパケットを受信できれば、元の情報パケットを復号することができる。この例の場合、送信パケットが5個で復号に必要なパケットの個数が7個である。このとき、余分に受け取るパケットの個数は2個になるので、パケットオーバーヘッドは2/5である。

良い噴水符号とは、実装が容易（すなわち、単純な操作で符号化・復号が可能）でオーバーヘッドの小さく、復号誤り率の低い符号である。筆者は文献(2)、(3)において、これらの条件を満たすような噴水符号を構成した。本稿の目的は、この噴水符号の符号化法・復号法並びに復号性能を紹介することにある。

本稿の構成とその役割は以下のとおりである。2.では、ブロック消失訂正符号の符号化法と復号法を簡単な例を用いて説明す

る。2.で与える例は3.以降で紹介する符号の基礎となっており、それらの理解に役立つ。3.では、既存の基礎的な噴水符号であるLT符号⁽⁴⁾とRaptor符号⁽⁵⁾の符号化法・復号法を与えると同時に、これらの噴水符号の問題点を明らかにする。4.では、文献(3)で与えている噴水符号の主要なアイデアであるシフト演算について紹介し、その効用を簡単な例を用いて示す。5.では、文献(3)で与えているシフト演算を利用した噴水符号の符号化法・復号法を紹介し、既存の噴水符号と性能を比較する。6.では、本稿を総括する。

2. ブロック消失訂正符号

本章では、ブロック消失訂正符号の簡単な例を幾つか紹介した後、ブロック消失訂正符号を記述するために必要な記法と定義を与える。

2.1 二 元 体

本稿では、送受信に用いる系列にビット列(0, 1の系列)を仮定している。ビット列に対する演算には二元体を用いる。二元体において、数の集合は{0, 1}であり、加算・乗算は次のように定められる。

$$0 + 0 = 0, \quad 0 + 1 = 1, \quad 1 + 0 = 1, \quad 1 + 1 = 0,$$

$$0 \times 0 = 0, \quad 0 \times 1 = 0, \quad 1 \times 0 = 0, \quad 1 \times 1 = 1.$$

$1 + 1 = 0$ 以外は整数の演算の結果と等しいことに注意せよ。二元体においては、 $-x = x$ が成立する^(注3)。すなわち、二元体における減算は加算として表現できる。

以降では、特に断りがない場合は、二元体による演算を仮定する。また、二元体を $\mathbb{F}_2$ と書き、二元体上の長さ ℓ のベクトルの集合を $\mathbb{F}_2^\ell$ と書くことにする。加えて、二元体上の $n \times k$ 行列の集合を $\mathbb{F}_2^{n \times k}$ と書く。

2.2 ブロック消失訂正符号の簡単な例

今、3シンボルから成るメッセージ(s_1, s_2, s_3)を伝送することを仮定する。以下では、2種類の符号化法を例示し、それぞれの性能についてまとめる。

例1. 第1の例は、符号化によって符号長4の符号語

$$(x_1, x_2, x_3, x_4) = (s_1, s_2, s_3, s_1 + s_2 + s_3) \quad (1)$$

を作成する方式である^(注4)。この例では、符号長が4でメッセージの長さが3であるので、符号化率が3/4になっている。

この符号化では、以降で示すとおり、一つまでの消失を必ず訂正することができる。消失を表すシンボルを*で書くことにする。例えば、1番目のシンボルが消失した場合は、受信者は $(*, x_2, x_3, x_4)$ を受け取る。この場合、 $x_4 = s_1 + s_2 + s_3$ で

(注3)： $-x = -x + (x + x) = (-x + x) + x = x$ から導出できる。

(注4)：本稿では、符号理論の慣例に従って、ベクトルは横ベクトルで表現する。

あったことに注意すると、

$$s_1 = x_4 - s_2 - s_3 = x_4 + s_2 + s_3 = x_4 + x_2 + x_3$$

が成り立つことが分かる^(注5)。したがって、 x_2, x_3, x_4 を元に、消失したシンボルである x_1 を復元できることが分かる。同様にして、2 番目のシンボルが消失した場合でも、3 番目のシンボルが消失した場合でも、消失したシンボルを復元可能であることが分かる^(注6)。

行列 $\mathbf{A}$ に対する転置行列を $\mathbf{A}^T$ で表すことにする。同様に、ベクトル $\mathbf{x}$ に対する転置を $\mathbf{x}^T$ と書く。式 (1) を行列を用いて表現すると、次のようになる。

$$(x_1, x_2, x_3, x_4)^T = \mathbf{A}(s_1, s_2, s_3)^T,$$

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}. \quad (2)$$

この行列を用いて、復号の手順を考えよう。今、受信者が $(*, x_2, x_3, x_4)$ を受信した場合を仮定する。行列 $\mathbf{A}$ の 2, 3, 4 行目から構成される行列を $\mathbf{A}_{(2,3,4)}$ と書くことにする。すると、式 (2) から、次の式が得られる。

$$(x_2, x_3, x_4)^T = \mathbf{A}_{(2,3,4)}(s_1, s_2, s_3)^T,$$

$$\mathbf{A}_{(2,3,4)} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

ここで、 $\mathbf{A}_{(2,3,4)}$ の階数 (ランク) が 3 であること、すなわちフルランクであること、に注意せよ。受信者は x_2, x_3, x_4 を得ているので、上記の方程式を解くことによって、メッセージ (s_1, s_2, s_3) を得ることができる。同様にして、 $(x_1, *, x_3, x_4)$, $(x_1, x_2, *, x_4)$, $(x_1, x_2, x_3, *)$ を受信した場合についても、 $\mathbf{A}_{(1,3,4)}$, $\mathbf{A}_{(1,2,4)}$, $\mathbf{A}_{(1,2,3)}$ のランクが 3 であるので、メッセージ (s_1, s_2, s_3) を得ることができる。したがって、この符号化を用いれば、一つまでの消失は必ず訂正することができることが分かる。

一方、 $\mathbf{A}$ から 2 行分削除した行列のランクは 2 であるので、消失を訂正することができない。例えば、 $(*, *, x_3, x_4)$ を受信した場合は、

$$(x_3, x_4)^T = \mathbf{A}_{(3,4)}(s_1, s_2, s_3)^T, \quad \mathbf{A}_{(3,4)} = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

のようになり、 $\mathbf{A}_{(3,4)}$ のランクが 2 であることから、 (s_1, s_2, s_3) を得ることができないことが分かる。

例 2. 第 2 の例は、符号化によって符号長 6 の符号語

$$(x_1, x_2, x_3, x_4, x_5, x_6) = (s_1, s_2, s_3, s_1 + s_2, s_2 + s_3, s_3 + s_1)$$

(注5)：第 2 等号では、二元体の性質である $-x = x$ を用いている。以降では、即座に $-x$ は $+x$ と変形する。

(注6)：4 番目のシンボルが消失した場合は、 $(s_1, s_2, s_3) = (x_1, x_2, x_3)$ であるから、復号せずとも元のメッセージが得られていることが分かる。

を作成する方式である。この例では、符号長が 6 でメッセージの長さが 3 であるので、符号化率が $1/2$ になっている。

この符号化では、以降で示すとおり、二つまでの消失を必ず訂正することができる。例 1 と同様に、この符号化を行列を用いて表現すると、次式を得る。

$$(x_1, x_2, x_3, x_4, x_5, x_6)^T = \mathbf{A}(s_1, s_2, s_3)^T,$$

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}.$$

この $\mathbf{A}$ から任意の 2 行を削除した行列のランクは必ず 3 になっていることが分かる。例えば、以下で与える行列のランクは全て 3 である。

$$\mathbf{A}_{(3,4,5,6)} = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}, \quad \mathbf{A}_{(2,3,5,6)} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix},$$

$$\mathbf{A}_{(1,2,3,6)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{pmatrix}.$$

したがって、この符号化は二つまでの消失を必ず訂正することが可能である。一方で、 $\mathbf{A}$ から 3 行を削除した行列のランクは 2 になる場合がある。例えば、 $\mathbf{A}_{(2,3,5)}$ のランクは 2 になる。したがって、この符号化では三つの削除を訂正することができない場合がある^(注7)。

以上の二つの例から、ブロック消失訂正符号の符号化率と消失訂正能力は行列 $\mathbf{A}$ の構成法に依存することが分かる。特に、訂正可能な消失の個数を増やすには符号化率を低くする必要があることが分かる。言い換えると、通信で発生する消失の割合に応じて適切な符号を用いなければ、情報伝達の効率が低減することが分かる。

2.3 線形符号

この節では、ブロック消失訂正符号を表現するために必要な記法と記号をまとめる。

前節の例によって、メッセージ $\mathbf{s} = (s_1, s_2, \dots, s_k)$ から符号語 $\mathbf{x} = (x_1, x_2, \dots, x_n)$ を作成するには、 $\mathbf{s}$ に行列 $\mathbf{A}$ を作用させればよいことが分かった。より正確には、 $\mathbf{x} = \mathbf{s}\mathbf{A}^T$ であることが分かった。今、 $\mathbf{G} := \mathbf{A}^T$ と定義することにする。この行列 $\mathbf{G}$ は生成行列と呼ばれる。符号語の集合を符号と呼び、生成行列 $\mathbf{G} \in \mathbb{F}_2^{k \times n}$ によって次のように定義される。

(注7)： $\mathbf{A}_{(1,2,5)}$ のランクは 3 なので、三つの削除を訂正ができる場合もある。

図2 式(5)で与えた $\mathbf{H}$ に対応するタナーグラフ
白丸は変数ノードを表し、白四角は検査ノードを表している。変数ノードは符号語のシンボルに対応しており、検査ノードは隣接する変数ノードのシンボルの和が0であることを表現している。

$$C := \{s\mathbf{G} \mid s \in \mathbb{F}_2^k\}. \quad (3)$$

符号 C は線形性 ($\mathbf{c}_1, \mathbf{c}_2 \in C \Rightarrow \mathbf{c}_1 + \mathbf{c}_2 \in C$) を満足するので、線形符号と呼ばれる。

符号 C は $\mathbb{F}_2^n$ の線形部分空間であるので、ある $n \times (n-k)$ 行列 $\mathbf{H}$ の核として表現可能である。すなわち、

$$C = \{x \in \mathbb{F}_2^n \mid \mathbf{H}x^T = \mathbf{0}^T \in \mathbb{F}_2^{n-k}\} \quad (4)$$

である。この行列 $\mathbf{H}$ をパリティ検査行列と呼ぶ。式(3), (4)から、 $\mathbf{H}$ と $\mathbf{G}$ は $\mathbf{H}\mathbf{G}^T = \mathbf{O} \in \mathbb{F}_2^{(n-k) \times k}$ を満たす。 $\mathbf{H}$ の (i, j) 成分を $h_{i,j}$ と書くことにする。

線形符号のグラフ表現の一つにタナーグラフ^(注8)が存在する。タナーグラフは2部グラフであり、 n 個の変数ノードと $n-k$ 個の検査ノード及びそれらを結ぶ辺で構成される。 $h_{i,j} = 1$ ならばそのときに限って i 番目の検査ノードと j 番目の変数ノードを接続する。 i 番目の検査ノードはパリティ検査行列の i 行目による制約 $\sum_{j=1}^n h_{i,j}x_j = 0$ を表現している。 j 番目の変数ノードは符号語の j 番目の要素 x_j の値を表現している。タナーグラフを利用すると復号の手順をグラフィカルに表現できる。

消失訂正能力の高い線形符号として LDPC 符号⁽⁸⁾が知られている。LDPC 符号は疎なパリティ検査行列 $\mathbf{H}$ で定義される線形符号である。特に、全ての列の1の個数が v で全ての行の1の個数が w の LDPC 符号を (v, w) 正則 LDPC 符号と呼ぶ。

例3. 例2で与えた符号に対して、そのパリティ検査行列 $\mathbf{H}$ は次式で与えられる。

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}. \quad (5)$$

このパリティ検査行列 $\mathbf{H}$ に対応するタナーグラフは図2のとおりである。図2の白丸は変数ノードを表し、白四角は検査ノードを表している。変数ノードは符号語のシンボルに対応しており、検査ノードは隣接する変数ノードのシンボルの和が0であることを表現している。例えば、1番目の検査ノードは1番目、2番目、4番目の変数ノードと隣接しており、 $x_1 + x_2 + x_4 = 0$ であることを表現している。同様に、2番目の検査ノードは $x_2 + x_3 + x_5 = 0$

(注8)：タナーグラフという名前は符号のグラフ表現を与えた R. Tanner⁽⁶⁾に由来する。ただし、今日のタナーグラフの定義でよく用いられるのは Wiberg⁽⁷⁾によって与えられたものである。

図3 復号で用いるグラフ 灰色の変数ノードは受信に成功したシンボルに対応している。

を表現し、3番目の検査ノードは $x_1 + x_3 + x_6 = 0$ を表現する。

次にタナーグラフ上での復号法を例を用いて説明しよう。いま、 $(*, *, x_3, x_4, x_5, x_6)$ が受信されたとする。正しく受信されたシンボルに対応する変数ノードとそれに接続する辺を灰色で描いたグラフが図3である。この図において、2番目と3番目の検査ノードは1本の黒い辺と接続している。これらの検査ノードでは、隣接しているシンボルの中で未知のものが一つだけであることを表現している。したがって、既知のシンボルを基にして未知のシンボルを復号可能であることを示している。2番目の検査ノードの制約は $x_2 + x_3 + x_5 = 0$ であったので、既知のシンボル x_3, x_5 を用いて未知のシンボル x_2 は $x_2 = x_3 + x_5$ によって復号すればよいことが分かる。同様に、3番目の検査ノードの制約は $x_1 + x_3 + x_6 = 0$ であったので、既知のシンボル x_3, x_6 を用いて未知のシンボル x_1 は $x_1 = x_3 + x_6$ によって復号すればよいことが分かる。このように、黒い辺と1本だけ接続している検査ノードを利用すれば復号が可能であることが分かる^(注9)。

3. 噴水符号

本節では、パケット通信におけるマルチキャストをモデル化し、そこで生じる問題を整理する。その後に噴水符号のコンセプトを紹介し、噴水符号の代表的かつ基礎的な構成法である LT 符号⁽⁴⁾と Raptor 符号⁽⁵⁾を紹介する。

3.1 問題設定

送信するメッセージが k 個のパケットに分割されていると仮定する。これらのパケットを情報パケットと呼び、 $\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_k$ で表すことにする。全ての情報パケットの長さは ℓ ビットであるとする。すなわち、 $\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_k \in \mathbb{F}_2^\ell$ である。

マルチキャストにおいて正確な情報伝達を達成するためには、全ての受信者が全ての情報パケットを復号できることが望まれる。また、受信者ごとに通信路消失率が異なるため、任意の通信路消失率に対応した符号である必要がある。加えて、スループットの観点から、受信すべきパケットの個数は少ない方がよく、復号時間は短い方がよい。

以上をまとめると、噴水符号に求められる性質は以下のとおりである。

- (1) 全ての情報パケットを高い確率で復号可能である（復

(注9)：この復号法が 3.3.2 で紹介するピーリング復号である。

号誤り率が低い).

- (2) 任意の通信路消失率に対応している.
- (3) 復号に必要なパケット数が少ない. 理想的には, 復号に必要なパケット数が k に等しい.
- (4) 復号計算量が小さい.

3.2 噴水符号のコンセプト

噴水符号においては, 送信者は情報パケットを符号化し, 十分大きな数の送信パケットを作成し, 送信パケットを送る. 受信者は送信パケットを一定の個数だけ受け取り, 情報パケットを復号する. 以降では, 復号に必要なパケット数を $\tilde{k}$ と書くこととする. このとき, 余分に受け取る送信パケットの割合であるパケットオーバーヘッド α は $\tilde{k} = k(1 + \alpha)$ を満たす. 噴水符号を用いたマルチキャストにおいては, 受信者が受け取った送信パケットを基に情報パケットを復号するため, 受信者は送信者にパケットの再送を要求する必要がある.

噴水符号を用いたマルチキャストの例については, 1. 及び図 1 を参照されたい.

3.3 LT 符号

噴水符号のコンセプトを初めて実現した符号が LT (Luby Transform) 符号⁽⁴⁾である. LT 符号では情報パケットをランダムに足して送信パケットを作成する. LT 符号はピーリング復号⁽⁹⁾によって $\mathcal{O}(k)$ ^(注10)の計算量で復号が可能である.

3.3.1 符号化

送信者は $\{1, 2, \dots, D\}$ 上の確率分布 $\Omega = (\Omega_1, \Omega_2, \dots, \Omega_D)$ をあらかじめ設計しておく^(注11). Ω は確率分布であるので, $\Omega_i \geq 0$ ($i = 1, 2, \dots, D$) と $\sum_{i=1}^D \Omega_i = 1$ を満足することに注意せよ. 確率分布 Ω は**次数分布**と呼ばれる. 送信パケットは以下の手続きによって作成される.

- (1) 次数分布 Ω に従って, 整数 d を選ぶ. すなわち, 確率 Ω_d で整数 d を選ぶ.
- (2) k 個の情報パケットから相異なる d 個を選ぶ. ここで選ばれたパケットを $\mathbf{s}_{i_1}, \mathbf{s}_{i_2}, \dots, \mathbf{s}_{i_d}$ と表すことにする.
- (3) 選ばれた d 個の情報パケットの和を送信パケットとして送る. すなわち, $\sum_{j=1}^d \mathbf{s}_{i_j}$ を送信パケットとして送る. なお, 加算した情報パケットのインデックス $(i_1, i_2, \dots, i_d)$ は送信パケットのヘッダに格納する.

以上の手続きを繰り返すことによって, 十分大きな数の送信パケットを作成する. 以降では, i 番目に作成した送信パケットを $\mathbf{x}_i$ と書き表すことにする.

LT 符号はファクタグラフを用いて図示することができる. ファクタグラフは情報パケットを表現する k 個の変数ノードと

図 4 LT 符号のファクタグラフ 白丸は変数ノードを表し, 黒四角はファクタノードを表している.

送信パケットを表現するファクタノードによって構成される. 変数ノードとファクタノードにはそれぞれ番号が付けられている. j 番目の送信パケットが $\mathbf{x}_j = \mathbf{s}_{i_1} + \mathbf{s}_{i_2} + \dots + \mathbf{s}_{i_d}$ によって作成された場合, j 番目のファクタノードは $i_1, i_2, \dots, i_d$ 番目の変数ノードと隣接する.

例 4. 今, 情報パケットの数が $k = 5$ で, 次数分布が $\Omega = (\Omega_1, \Omega_2, \Omega_3) = (1/3, 1/2, 1/6)$ で与えられると仮定する. 符号化の手続きに従って, 1 番目の送信パケットを作成しよう. 第 1 ステップで, d として 3 が選ばれたとする. そして, 第 2 ステップで, $\mathbf{s}_1, \mathbf{s}_3, \mathbf{s}_4$ が選択されたとする. この場合, 第 3 ステップでは $\mathbf{x}_1 = \mathbf{s}_1 + \mathbf{s}_3 + \mathbf{s}_4$ を送信パケットとして送信する. 同様の手順によって, 2 番目以降の送信パケットも作成する. ここでは, 次のように以降の送信パケットが作成されたものとする.

$$\begin{aligned} \mathbf{x}_1 &= \mathbf{s}_1 + \mathbf{s}_3 + \mathbf{s}_4, & \mathbf{x}_2 &= \mathbf{s}_1 + \mathbf{s}_2, & \mathbf{x}_3 &= \mathbf{s}_2 + \mathbf{s}_4, \\ \mathbf{x}_4 &= \mathbf{s}_2, & \mathbf{x}_5 &= \mathbf{s}_3 + \mathbf{s}_4, & \mathbf{x}_6 &= \mathbf{s}_3, \\ \mathbf{x}_7 &= \mathbf{s}_1 + \mathbf{s}_5, & \mathbf{x}_8 &= \mathbf{s}_3 + \mathbf{s}_4, & \dots \end{aligned}$$

図 4 は以上の符号化を表現するファクタグラフを与えている. 白丸は変数ノードを表し, 黒四角はファクタノードを表している.

3.3.2 復号

噴水符号においては, 受信者ごとに受信されるパケットが異なる. LT 符号の復号では, まずはじめに, 受信者は受信したパケットのヘッダを基にファクタグラフを構成する. その後に, 作成したファクタグラフ上でピーリング復号法⁽⁹⁾と呼ばれるメッセージパッシングアルゴリズム^(注12)を用いて復号をする. ピーリング復号法はガウスの消去法の後退代入であるともみなすことができる. この項では, まずはじめに LT 符号の復号法を例を通じて説明し, その後に復号アルゴリズムをまとめる.

例 5. 例 4 のように送信パケットが作成されたとする. ある受信者は 1 番目から 7 番目の送信パケットを受信できたと仮定する. この例では, この受信者の復号手順を示す.

LT 符号の復号では, まずファクタグラフを作成する. 1 番目に受信したパケットのヘッダ情報から, 1 番目の送信パケット $\mathbf{x}_1$ は $\mathbf{x}_1 = \mathbf{s}_1 + \mathbf{s}_3 + \mathbf{s}_4$ を満たすことが分かる. したがって, 1 番目のファクタノードは 1 番目と 3 番目と 4 番目の変数ノード

(注10): (k に依存する) ある量 $g(k)$ が $\mathcal{O}(f(k))$ であるとは, (大まかに言う) 十分大きな k に対して, $g(k)$ が $f(k)$ に比例することを表す. より正確には, ある定数 b を用いて, $g(k) \leq bf(k)$ と表せることを指す.

(注11): Ω の設計法は 3.3.3 で解説する.

(注12): メッセージパッシングアルゴリズムとは, グラフ上で動作するアルゴリズムの一つである. このアルゴリズムでは, ノード同士が辺を介してメッセージをやり取りする. 各ノードは受け取ったメッセージを基にして, 次に送るメッセージを計算する. 以上の手続きを何度か繰り返すことで, 所望の計算結果を得る.

ドと接続すればよいことが分かる。2 番目に受信したパケットのヘッダ情報から、2 番目の送信パケット $\mathbf{x}_2$ は $\mathbf{x}_2 = \mathbf{s}_1 + \mathbf{s}_2$ を満たすことが分かる。したがって、2 番目のファクタノードは 1 番目と 2 番目の変数ノードと接続すればよい。同様にして、受信した送信パケットのヘッダ情報から図 5(a) のようなファクタグラフを得ることが分かる。

次に作成したファクタグラフを基に復号をしよう。図 5(a)を見ると、4 番目のファクタノードは次数が 1 であることが分かる。4 番目のファクタノードは 2 番目の変数ノードとのみ接続しているため、 $\mathbf{x}_4 = \mathbf{s}_2$ を表現している。すなわち、4 番目の受信パケットから 2 番目の情報パケット $\mathbf{s}_2$ を復号可能であることが分かる。2 番目の変数ノードは 2 番目と 3 番目のファクタノードとも隣接していることから、これらの対応する受信パケットには $\mathbf{s}_2$ が加算されている。 $\mathbf{s}_2$ は既知なので、これらの受信パケットに $\mathbf{s}_2$ を加えることによって、

$$\mathbf{x}_2 \leftarrow \mathbf{x}_2 + \mathbf{s}_2 = (\mathbf{s}_1 + \mathbf{s}_2) + \mathbf{s}_2 = \mathbf{s}_1$$

$$\mathbf{x}_3 \leftarrow \mathbf{x}_3 + \mathbf{s}_2 = (\mathbf{s}_2 + \mathbf{s}_4) + \mathbf{s}_2 = \mathbf{s}_4$$

と情報を更新することができる^(注13)。この情報の更新によって新たに得られるファクタグラフを図 5(b) に与える。図 5(a) と図 5(b) を見比べると、2 番目の変数ノードとそれに隣接した辺がなくなっていることが分かる。

同様に、図 5(b) を見ると、2 番目のファクタノードの次数が 1 であり、 $\mathbf{x}_2 = \mathbf{s}_1$ を表現している。したがって、1 番目の情報パケットを 2 番目のファクタノードから復号可能である。その結果から $\mathbf{x}_1, \mathbf{x}_7$ に $\mathbf{s}_1$ を加えることによって、次のような更新がなされる。

$$\mathbf{x}_1 \leftarrow \mathbf{x}_1 + \mathbf{s}_1 = (\mathbf{s}_1 + \mathbf{s}_3 + \mathbf{s}_4) + \mathbf{s}_1 = \mathbf{s}_3 + \mathbf{s}_4$$

$$\mathbf{x}_7 \leftarrow \mathbf{x}_7 + \mathbf{s}_1 = (\mathbf{s}_1 + \mathbf{s}_5) + \mathbf{s}_1 = \mathbf{s}_5$$

この情報の更新によって新たに得られるファクタグラフを図 5(c) に与える。

同様に、復号を進めていくと、図 5(d), 5(e), 5(f) のように復号が進んでいき、全ての情報パケットを復号できることが分かる。

以上の例をまとめると、ピーリング復号の動作は次のようにまとめられる。

- (1) 次数 1 のファクタノードを探し、隣接する変数ノード v を復号する。
- (2) 変数ノード v に隣接するファクタノードから復号結果を引いて、変数ノード v とそれに隣接する全ての辺を削除する。
- (3) 次数 1 のファクタノードが存在すれば、step 1 に戻る。そうでなければ復号を終了する。

復号終了時点において、全ての辺がなくなっていれば、すなわち全ての変数ノードが復号されていれば、復号成功である。

ピーリング復号の計算量はファクタグラフ中の辺の総数 ξ に比例する。ファクタノードの次数の期待値を $\bar{\Omega}$ で表すことにす

(注13)：本稿では、変数 a の値を b に更新することを $a \leftarrow b$ と書く。

図 5 LT 符号に対するピーリング復号 白丸は変数ノードを表し、黒四角はファクタノードを表している。復号によって削除されたノードと辺は灰色で表されている。

ると、 $\bar{\Omega} = \sum_{i=1}^D i\Omega_i$ を満たす。ファクタグラフ中の辺の総数の期待値は $\bar{\xi} = \bar{k}\bar{\Omega} = k(1 + \alpha)\bar{\Omega}$ である。したがって、 $\bar{\Omega}$ と α が k に依存しない場合、計算量は $\mathcal{O}(k)$ であることが分かる。

3.3.3 次数分布の設計と LT 符号の問題点^(注14)

前項まででは、次数分布 Ω はうまく設計済みであることを仮定していた。本項では、次数分布 Ω の設計指針と LT 符号の問題点を紹介する。

3.3.2 で紹介したピーリング復号を開始するには次数 1 のファクタノードが必要であった。したがって、 $\Omega_1 > 0$ を満たしていなければならない。更に、3.3.2 では、ピーリング復号の計算量はファクタグラフ中の辺の総数 ξ に比例することを説明し、 $\bar{\xi} = \bar{k}\bar{\Omega}$ であることを示した。また、パケットオーバーヘッド並びに復号誤り率も Ω に依存する。以上をまとめると、次数分布 Ω は次の条件を満たす必要がある。

- $\Omega_1 > 0$ であること。
- $\bar{k}\bar{\Omega}$ が小さいこと、理想的には $\bar{k}\bar{\Omega} = \mathcal{O}(k)$ であることが望ましい。
- パケットオーバーヘッド並びに復号誤り率の低い Ω であること。

しかしながら、以降に述べるとおり、これらの条件を同時に満たすことが困難であることが知られている。

まずは、復号が失敗するような状況を考えよう。LT 符号では、ランダムに選んだ情報パケットを足し合わせて送信パケットと

(注14)：この節は専門的な内容を含む。加えて、本節の要点は 3.4 の冒頭にまとめている。したがって、噴水符号に対して深い関心がある読者以外はこの節を読み飛ばして構わない。

図 6 LT 符号の復号が失敗してしまう例 この例では 5 番目の情報パッケージがどの送信パッケージにも接続していないので、復号することができない。なお、この図は例 4 において、1 番目から 6 番目及び 8 番目の送信パッケージを受け取った状況に対応している。

して送信する。その結果、受信側でファクタグラフを作成した際に、図 6 の 5 番目の変数ノードのように、どのファクタノードにも接続していない変数ノードが発生し得る。すなわち、復号をしたとしても消失したままの情報パッケージが発生してしまう。このように LT 符号はランダム性を利用している符号であるので、ある確率で一部の情報パッケージに関する情報を受け取れなくなってしまう。その結果として、LT 符号は復号誤り率が高くなりやすい。また、復号誤り率を低くするには、次の命題が示すとおり、復号計算量を犠牲にする必要がある。

命題 1. (5, Proposition 1) 情報パッケージ数が k 個の LT 符号で復号が成功するためには、ファクタグラフ中に少なくとも $O(k \log(k))$ 本の辺が必要である。

ファクタグラフ中の辺の総数の期待値が $k(1 + \alpha)\bar{\Omega}$ であったので、小さな α を達成するには $\bar{\Omega}$ を大きくし、復号計算量を犠牲にする必要があることが分かる。

3.4 Raptor 符号

3.3.3 では、LT 符号の復号性能の劣化の原因はどのファクタノードにも接続しない孤立した変数ノードの存在であることを説明し、オーバーヘッドの小さな LT 符号を構成するには復号計算量を犠牲にする必要があることを紹介した。本節で紹介する Raptor 符号はこの問題を解消し、 $O(k)$ の計算量で復号可能で小さなパッケージオーバーヘッドを達成した。Raptor 符号では事前符号化と呼ばれる操作を LT 符号に施すことによって、前述の問題点を解消している。事前符号化と呼ばれる処理では、情報パッケージに対してブロック消失訂正符号を適用し、事前符号化パッケージを作成する。作成された事前符号化パッケージに対して LT 符号を適用することで、送信パッケージを作成する。LT 符号側で復号できない事前符号化パッケージが存在したとしても、ブロック消失訂正符号の復号を施すことによって、そのパッケージを復号できるようになる。

3.4.1 符号化法

送信者は事前符号 C と次数分布 Ω をあらかじめ設計しておく^(注15)。利用する事前符号 C は全ての受信者にあらかじめ伝え

(注15)：事前符号には符号化率の高いブロック消失訂正符号（例えば、LDPC 符号）が用いられる。

ておく。送信者は次のようにして送信パッケージを作成する。

- (1) k 個の情報パッケージ $s_1, s_2, \dots, s_k$ を事前符号で符号化し、 n 個の事前符号化パッケージ $a_1, a_2, \dots, a_n$ を作成する。
- (2) 作成した事前符号化パッケージに対して LT 符号化を施すことで十分大きな数の送信パッケージを作成する。すなわち、以下の手続きを繰り返すことで、送信パッケージを作成する。
 - (a) 次数分布 Ω に従って、整数 d を選ぶ。
 - (b) n 個の事前符号化パッケージから相異なる d 個を選ぶ。ここで選ばれたパッケージのインデックスを $(i_1, i_2, \dots, i_d)$ と表すことにする。
 - (c) 選ばれた d 個の事前符号化パッケージの和を送信パッケージとして送る。すなわち、 $\sum_{j=1}^d a_{i_j}$ を送信パッケージとして送る。なお、加算した事前符号化パッケージのインデックス $(i_1, i_2, \dots, i_d)$ は送信パッケージのヘッダに格納する。

例 6. 事前符号 C がパリティ検査行列 $H = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \end{pmatrix}$ で定義される場合を考える。このとき、 $k = 4$ で $n = 5$ である。パリティ検査行列 H から、事前符号化パッケージは $a_1 + a_2 + a_3 + a_4 + a_5 = 0$ を満たす。したがって、情報パッケージ s_1, s_2, s_3, s_4 を $a_1 = s_1, a_2 = s_2, a_3 = s_3, a_4 = s_4$ と対応付ければ、5 番目の事前符号化パッケージは $a_5 = s_1 + s_2 + s_3 + s_4$ で作成できることが分かる。

事前符号化パッケージに対して LT 符号化を施すことによって、次のような送信パッケージを得たとする。

$$\begin{aligned} x_1 &= a_1 + a_3 + a_4, & x_2 &= a_1 + a_2, & x_3 &= a_2 + a_4, \\ x_4 &= a_2, & x_5 &= a_3 + a_4, & x_6 &= a_3, \\ x_7 &= a_1 + a_5, & x_8 &= a_3 + a_4, & \dots \end{aligned}$$

このときのファクタグラフは図 7 のように与えられる。このファクタグラフは事前符号のタナグラフと LT 符号のファクタグラフを接続した形状になっている。すなわち、ファクタグラフの上部は事前符号のタナグラフに対応しており、下部は LT 符号に対応している。図 7 において、白丸は事前符号化パッケージに、白四角は事前符号の検査ノードに、黒四角は送信パッケージにそれぞれ対応している。

3.4.2 復号法

Raptor 符号の復号においては、まずはじめに、受信者は事前符号のタナグラフと受信したパッケージのヘッダを基にしてファクタグラフを構成する。その後、作成したファクタグラフ上でピーリング復号をする。

例 7. 例 6 のように送信パッケージが作成されたとする。ある受信者は 1 番目から 5 番目の送信パッケージを受信できたと仮定する。このとき、情報パッケージ数が 4 で受信したパッケージ数が 5 であるので、パッケージオーバーヘッドは $1/4$ である。この例では、この受信者の復号手順を示す。

受信者はまずはじめにファクタグラフを作成する。作成した

図 7 Raptor 符号のファクタグラフ 白丸は変数ノードを表し、白四角は事前符号の検査ノードを表し、黒四角はファクタノードを表している。

ファクタグラフは図 8(a) のようになる。

次に作成したファクタグラフを基に復号をしていこう。図 8(a)を見ると、4 番目のファクタノードは次数が 1 であることが分かる。4 番目のファクタノードは 2 番目の変数ノードとのみ接続しているので、 $x_4 = a_2$ を表現している。すなわち、4 番目の受信パケットから 2 番目の事前符号化パケット a_2 を復号可能であることが分かる。 a_2 は既知なので、これらの受信パケットに a_2 を加えることによって、

$$x_2 \leftarrow x_2 + a_2 = (a_1 + a_2) + a_2 = a_1$$

$$x_3 \leftarrow x_3 + a_2 = (a_2 + a_4) + a_2 = a_4$$

と情報を更新することができる。この情報の更新によって新たに得られるファクタグラフを図 8(b) に与える。図 8(a) と図 8(b) を見比べると、2 番目の変数ノードとそれに隣接した辺がなくなっていることが分かる。

同様に、復号を進めていくと、図 8(c), 8(d), 8(e), 8(f) のように復号が進んでいき、全ての事前符号化パケットを復号できることが分かる。

3.4.3 問題点

受信パケットによって構成されるファクタグラフが図 9(a) のように与えられる場合を考えよう。このとき、ピーリング復号を進めていくと、図 9(b) のような状態で復号が停止してしまう。すなわち、次数 1 の検査ノードあるいはファクタノードが存在しなくなると復号が停止してしまう。

したがって、パケットオーバーヘッドと復号誤り率を改善するには、何らかの方法でこのようなピーリング復号の復号停止状態に陥っても復号が続けられる仕組みを導入する必要がある。本稿の主題はこのような仕組みに次章で紹介するシフト演算を利用することである。

4. シフト演算とその効用

この章では、シフト演算を利用したブロック消失訂正符号について簡単な例を通じて説明し、以降の節で必要な記号を導入する。

図 8 Raptor 符号に対するピーリング復号の例 灰色ノードと辺は削除されたことを表す。

図 9 Raptor 符号において復号が失敗する例

4.1 シフト演算を利用したブロック消失訂正符号の例

この節ではシフト演算を利用したブロック消失訂正符号^{(10), (11)}の符号化と復号を簡単な例を通じて説明する。

例 8. 今、三つの情報パケット s_1, s_2, s_3 を伝送することを考える。それぞれの情報パケットの長さは l であるとする。したがって、 s_i は $(s_{i,1}, s_{i,2}, s_{i,3}, \dots, s_{i,l})$ のように長さ l のベクトルによって表現される。

この例では符号化によって五つの符号化パケット x_1, x_2, x_3, x_4, x_5 を作成する。はじめの三つの符号化パケットは $x_1 = s_1$,

$$\begin{array}{r}
\begin{array}{cccccc} s_{1,1} & s_{1,2} & s_{1,3} & \cdots & s_{1,\ell-1} & s_{1,\ell} \\ s_{2,1} & s_{2,2} & s_{2,3} & \cdots & s_{2,\ell-1} & s_{2,\ell} \end{array} \\
+ \\
\hline
\begin{array}{cccccc} y_{4,1} & y_{4,2} & y_{4,3} & \cdots & y_{4,\ell-1} & y_{4,\ell} \end{array}
\end{array}$$

(a) y_4 が満たす条件式. $y_{4,i} = s_{1,i} + s_{2,i}$ を満たしている

$$\begin{array}{r}
\begin{array}{cccccc} s_{1,1} & s_{1,2} & s_{1,3} & \cdots & s_{1,\ell-1} & s_{1,\ell} & 0 & 0 \\ 0 & s_{2,1} & s_{2,2} & \cdots & s_{2,\ell-2} & s_{2,\ell-1} & s_{2,\ell} & 0 \end{array} \\
+ \\
\hline
\begin{array}{cccccc} y_{5,1} & y_{5,2} & y_{5,3} & \cdots & y_{5,\ell-1} & y_{5,\ell} & y_{5,\ell+1} & 0 \end{array}
\end{array}$$

(b) y_5 が満たす条件式. $y_{5,1} = s_{1,1}$, $y_{5,i} = s_{1,i} + s_{2,i-1}$ ($i = 2, 3, \dots, \ell$), $y_{5,\ell+1} = s_{2,\ell}$ を満たしている.

図 10 y_4, y_5 が満たす条件式

$\mathbf{x}_2 = \mathbf{s}_2$, $\mathbf{x}_3 = \mathbf{s}_3$ のように作成する. 4 番目の符号化パケットでは, 三つの情報パケットの和を用いて作成する. すなわち, $\mathbf{x}_4 = \mathbf{s}_1 + \mathbf{s}_2 + \mathbf{s}_3$ である. 5 番目の符号化パケットでは, 以下のとおりに, $\mathbf{s}_1$ と $\mathbf{s}_2$ を一つ右シフトしたものと $\mathbf{s}_3$ を二つ右シフトしたものの和で作成する.

$$\begin{aligned}
& (s_{1,1}, s_{1,2}, s_{1,3}, s_{1,4}, \dots, s_{1,\ell}, 0, 0) \\
& + (0, s_{2,1}, s_{2,2}, s_{2,3}, \dots, s_{2,\ell-1}, s_{2,\ell}, 0) \\
& + (0, 0, s_{3,1}, s_{3,2}, \dots, s_{3,\ell-2}, s_{3,\ell-1}, s_{3,\ell}) \\
& = (x_{5,1}, x_{5,2}, x_{5,3}, x_{5,4}, \dots, x_{5,\ell}, x_{5,\ell+1}, x_{5,\ell+2}).
\end{aligned}$$

符号化パケット $\mathbf{x}_5$ は長さ $\ell+2$ のベクトルになっており, 他の符号化パケットと比べて 2 ビット分だけ長い. ただし, ℓ が十分に長いときは, このパケットの長さの増加はほとんど無視できる (注16).

この符号は任意の三つの符号化パケットから情報パケットを復号することができる. この例では, $\mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5$ から情報パケットを復号する例を考えよう. $\mathbf{x}_3 = \mathbf{s}_3$ であつたので, 3 番目の情報パケットは直ちに復号できる. $\mathbf{s}_3$ が復号できたので, $\mathbf{y}_4 = \mathbf{x}_4 + \mathbf{s}_3$, $\mathbf{y}_5 = \mathbf{x}_5 + (0, 0, s_{3,1}, s_{3,2}, \dots, s_{3,\ell})$ を計算すると, $\mathbf{y}_4, \mathbf{y}_5$ は図 10(a), 10(b) のように与えられる.

新たに得られた $\mathbf{y}_4, \mathbf{y}_5$ から $\mathbf{s}_1, \mathbf{s}_2$ が復号可能であることを示していこう. 図 10(b) から, $s_{1,1} = y_{5,1}$ であることが分かる. したがって, $y_{5,1}$ を基に 1 番目の情報パケットの 1 ビット目 $s_{1,1}$ が復号可能である. 次に, 図 10(a) から, $y_{4,1} = s_{1,1} + s_{2,1}$ であることが分かる. $s_{1,1}$ は復号済みであるので, $y_{4,1} + s_{1,1} = s_{2,1}$ から, $s_{2,1}$ を復号可能である. 更に図 10(b) から, $y_{5,2} = s_{1,2} + s_{2,1}$ が分かる. $s_{2,1}$ は復号済みであるので, $y_{5,2} + s_{2,1} = s_{1,2}$ によって, $s_{1,2}$ を復号可能である. 同様にすれば, $\mathbf{s}_1, \mathbf{s}_2$ の全てのビットを復号することが可能である. 図 11 は, 以上の復号の順番を $\mathbf{y}_4, \mathbf{y}_5$ の上に赤線で書き込んだものである. この図から, 復号の順序はジグザグと動いていることが分かる. この動作にちなんで, この復号法はジグザグ復号法⁽¹²⁾と呼ばれている.

(注16) : イーサネット (有線 LAN のプロトコル) の場合, パケットの長さはおおよそ 1,000 Byte (8,000 bit) である. したがって $\ell \approx 8,000$ であるので, パケットの長さの増加は相対的に僅かである.

$$\begin{array}{r}
\begin{array}{cccccc} s_{1,1} & s_{1,2} & s_{1,3} & \cdots & s_{1,\ell-1} & s_{1,\ell} \\ s_{2,1} & s_{2,2} & s_{2,3} & \cdots & s_{2,\ell-1} & s_{2,\ell} \end{array} \\
+ \\
\hline
\begin{array}{cccccc} y_{4,1} & y_{4,2} & y_{4,3} & \cdots & y_{4,\ell-1} & y_{4,\ell} \end{array} \\
+ \\
\hline
\begin{array}{cccccc} s_{1,1} & s_{1,2} & s_{1,3} & \cdots & s_{1,\ell-1} & s_{1,\ell} & 0 & 0 \\ 0 & s_{2,1} & s_{2,2} & \cdots & s_{2,\ell-2} & s_{2,\ell-1} & s_{2,\ell} & 0 \end{array} \\
+ \\
\hline
\begin{array}{cccccc} y_{5,1} & y_{5,2} & y_{5,3} & \cdots & y_{5,\ell-1} & y_{5,\ell} & y_{5,\ell+1} & 0 \end{array}
\end{array}$$

図 11 ジグザグ復号の復号順序を $\mathbf{y}_4, \mathbf{y}_5$ 上に赤線で書き込んだ図. 復号順序が $\mathbf{y}_4, \mathbf{y}_5$ の上をジグザグな経路をたどることが分かる.

4.2 符号化と復号法

4.2.1 記法と符号化

シフト演算の表現を用意するために, ベクトルに対する多項式表現を導入する. ベクトル $\mathbf{s}_i = (s_{i,1}, s_{i,2}, \dots, s_{i,\ell})$ に対する多項式表現を $s_i(z) = \sum_{j=1}^{\ell} s_{i,j} z^{j-1}$ で与える. すなわち

$$\begin{aligned}
\mathbf{s}_i &= (s_{i,1}, s_{i,2}, s_{i,3}, \dots, s_{i,\ell}) \\
s_i(z) &= s_{i,1} + s_{i,2}z + s_{i,3}z^2 + \cdots + s_{i,\ell}z^{\ell-1}
\end{aligned}$$

である. 多項式表現を利用すると $s_i(z)$ に対する t 回の右シフトが $z^t s_i(z)$ によって表現可能である. したがって, 符号化パケットに対する多項式表現 $x_j(z)$ は

$$x_j(z) = \sum_{i=1}^k g_{i,j} s_i(z)$$

のように単項式 $g_{i,j} \in \{0, 1, z, z^2, \dots\}$ を利用して表現が可能である. 行列 $\mathbf{G}$ を $\mathbf{G} = (g_{i,j})$ と定めると,

$$(x_1(z), x_2(z), \dots, x_n(z)) = (s_1(z), s_2(z), \dots, s_k(z))\mathbf{G}$$

と表現できる. したがって, 行列 $\mathbf{G}$ を利用すれば符号の表現が可能である.

この符号に対応するファクタグラフ $\mathbf{G}$ を記述しよう. このファクタグラフには k 個の変数ノードと n 個のファクタノード及びラベル付き辺によって構成される. 各変数ノードは情報パケットに対応しており, 各ファクタノードは符号化パケットに対応している. $g_{i,j} = z^t$ のときに i 番目の変数ノードと j 番目のファクタノードが z^t でラベル付けされた辺で接続される.

例 9. 例 8 で与えた符号化を多項式によって表現すると

$$\begin{aligned}
x_1(z) &= s_1(z), & x_4(z) &= s_1(z) + s_2(z) + s_3(z), \\
x_2(z) &= s_2(z), & x_5(z) &= s_1(z) + z s_2(z) + z^2 s_3(z), \\
x_3(z) &= s_3(z),
\end{aligned} \quad (6)$$

のように与えられる. これらの式から行列 $\mathbf{G}$ は

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & z \\ 0 & 0 & 1 & 1 & z^2 \end{pmatrix}$$

で与えられる. この符号化に対応するファクタグラフは図 12 の

図 12 式 (6) に対応するファクタグラフ 辺に付いているラベルはシフト量を表しており、 z^t は t 回右シフトを施すことを意味している。なお、ラベル 1 は省略している。

とおりである。

4.2.2 復 号

復号に用いるファクタグラフ G' は G の部分グラフであり、変数ノードと受信に成功したパケットに対応するファクタノードとそれらに接続する辺から構成される。復号においては、まずはじめに、 G' 上でピーリング復号をする。このピーリング復号が停止したら、残ったグラフ (残余グラフ) に対して次の操作を行うことによって G_b を作成する。

- (1) i 番目の変数ノード v_i を ℓ 個の変数ノード $v_{i,1}, v_{i,2}, \dots, v_{i,\ell}$ に変換する。
- (2) j 番目のファクタノード f_j に対応する符号化パケットの長さを ℓ_j で表す。このとき、 j 番目のファクタノード f_j を ℓ_j 個のファクタノード $f_{j,1}, f_{j,2}, \dots, f_{j,\ell_j}$ に変換する。
- (3) v_i と f_j がラベル z^t の辺で接続されているとき、 $k \in \{1, 2, \dots, \ell\}$ に対して $v_{i,k}$ と $f_{j,k+t}$ を辺で接続する。

得られた G_b に対してピーリング復号を行うことによって更に復号を進める。ピーリング復号によって G_b の全ての辺が削除されれば、復号が成功する。以降では G_b のことを残余グラフのビット表現と呼ぶ。また、 G' 上でのピーリング復号をパケットごとのピーリング復号と呼び、 G_b 上でのピーリング復号をビットごとのピーリング復号と呼ぶことにする。

例 10. 例 9 のように符号化がなされたと仮定する。いま、受信者が $x_3(z), x_4(z), x_5(z)$ を受信できたとしよう。このとき、受信者が作成するファクタグラフ G' は図 13 のとおりである。このファクタグラフ上でピーリング復号をすると図 14 が得られる。この状況は、例 8 において、 y_4, y_5 を計算した状況に対応している。図 14 には次数 1 のファクタノードが存在しないのでパケットごとのピーリング復号が停止する。次に、図 14 に対応する残余グラフのビット表現 G_b を構成する。得られた G_b は図 15 のとおりになる。 G_b 上でビットごとのピーリング復号を行うと、 $s_{1,1}, s_{2,1}, s_{1,2}, s_{2,2}$ の順に復号が進んでいく。この復号手順は、例 8 における、 y_4, y_5 から s_1, s_2 を復号する手順に対応している。

以上の例から、シフト演算を利用すれば、パケットごとのピーリング復号で復号が停止してもビットごとのピーリング復号によって復号が可能な場合がある。したがって、噴水符号にシフト演算を適用すれば、Raptor 符号の問題点を解消し、更なる性

図 13 復号に用いるファクタグラフ G'

図 14 パケットごとのピーリング復号終了時の残余グラフ

図 15 残余グラフのビット表現 G_b 灰色の枠は残余グラフにおいて一つのノードであったことを表している。

能向上が見込める。

4.3 シフトを利用した関連研究^(注17)

本節では、シフト操作を利用したブロック消失訂正符号に関する研究を紹介する。

Gollakota と Katabi⁽¹²⁾ は無線通信における隠れ端末問題を解消するために、衝突の生じた二つの受信信号から送られてきた二つのパケットを復元する手法を考えた。この手法では、衝突した受信信号において、そのタイミングが僅かにずれていることを利用して、ジグザグ復号をすることで元のパケットを復元している。Sung と Gong⁽¹⁰⁾ はジグザグ復号が可能な符号をシフト演算を利用することで構成し、計算量の小さな分散ストレージシステムを構築した。筆者⁽²⁾ は 4.2.2 で紹介した事項を示した、すなわちシフト演算を利用した符号に対するファクタグラフ表現を与え、ジグザグ復号がビットごとのピーリング復号とみなせることを示した。更に筆者⁽²⁾ は以上の事実を利用してシフト演算を利用した噴水符号を構成し、その性能を明らかにした。

本節で解説した事項とは外れるが、シフトを利用した消失訂正符号に関する既存研究をいくつか紹介していく。Al-Shaikhhi と Ilow⁽¹³⁾ は 2008 年にシフト演算を利用したブロック消失

(注17)：初学者はこの節を読み飛ばしてよい。

訂正符号に最ゆう (ML) 復号法^(注18)を適用した。Hanaki と筆者⁽¹⁴⁾はシフトを利用した消失訂正符号に対する ML 復号法が並列計算可能であることを示して、文献(13)の結果よりも冗長性が低い符号の構成法を与えた。Qureshi ら⁽¹⁵⁾はネットワーク符号化に対して、Fu ら⁽¹⁶⁾は再生成符号に対して、Gong ら⁽¹⁷⁾は秘密分散法に対して、シフト演算を利用した符号を利用している。また、ランダムアクセスプロトコルに対するジグザグ復号の適用として、Ogata と Ishibashi の研究⁽¹⁸⁾、⁽¹⁹⁾、Oinaga らの研究⁽²⁰⁾、Emoto と筆者の研究⁽²¹⁾が存在する。

5. シフト演算を利用した噴水符号

シフト演算を利用した噴水 (ZDF: Zigzag Decodable Fountain) 符号は Raptor 符号にシフト演算を組み合わせた符号である。復号はパケットごとのピーリング復号とビットごとのピーリング復号を組み合わせることで行われる。この章では、ZDF 符号の符号化法・復号法をまとめ、数値実験によって性能を評価する。

5.1 符号化法

送信者は事前符号 C と次数分布 Ω とシフト分布 $\Delta = (\Delta_0, \Delta_1, \dots, \Delta_{D_m})$ をあらかじめ設計しておく。なお、シフト分布は確率分布を表しており、 $\sum_{i=0}^{D_m} \Delta_i = 1$ を満たす。利用する事前符号は全ての受信者にあらかじめ伝えておく。送信者は次のようにして送信パケットを作成する。

- (1) k 個の情報パケット $s_1, s_2, \dots, s_k$ を事前符号で符号化し、 n 個の事前符号化パケット $a_1, a_2, \dots, a_n$ を作成する。
- (2) 次の手続きを繰り返すことで送信パケットを作成する。
 - (a) 次数分布 Ω にしたがって、整数 d を選ぶ。
 - (b) n 個の事前符号化パケットから相異なる d 個を選ぶ。ここで選ばれたパケットのインデックスを $(i_1, i_2, \dots, i_d)$ と表すことにする。
 - (c) シフト分布に従って d 個の整数 $(\delta_1, \delta_2, \dots, \delta_d)$ を独立に選ぶ。すなわち、任意の $i \in \{1, 2, \dots, d\}$ に対して確率 Δ_k で $\delta_i = k$ にする。
 - (d) 次で与えられる多項式表現を送信パケットとして送信する。

$$\sum_{j=1}^d z^{\delta_j} a_{i_j}(z)$$

なお、 $(i_1, i_2, \dots, i_d)$ 及び $(\delta_1, \delta_2, \dots, \delta_d)$ は送信パケットのヘッダに格納しておく。

注意 1. シフト分布を $\Delta_0 = 1$ にすると、送信パケットは事前符号化パケットの和として表現される。したがって、ZDF 符号は Raptor 符号の一般化になっている。

(注18)：消失訂正をする際にガウスの消去法を利用する手法と等価な復号法である。復号性能が良いが、復号計算量が大きいことが知られている。

5.2 復号法

第1に、受信者は事前符号のタナグラフと受信したパケットのヘッダを基にしてファクタグラフを構成する。第2に、作成したファクタグラフ上でパケットごとのピーリング復号をする。パケットごとのピーリング復号に失敗した場合はその残余グラフのビット表現 G_b を作成する^(注19)。最後に、作成した G_b 上でビットごとのピーリング復号をする。

5.3 性能評価

本節では、数値実験によって Raptor 符号と ZDF 符号の性能を比較する。

5.3.1 評価指標

噴水符号の評価指標にはパケットオーバーヘッド α によって評価されていた。パケットオーバーヘッド α は情報パケット数と余分に受け取ったパケット数の比であり、情報パケット数が k で受信したパケット数が $\tilde{k}$ のときに

$$\alpha = \frac{\tilde{k} - k}{k}$$

で計算できる。

一方、ZDF 符号では、符号化によって送信パケットの長さが僅かに長くなる。したがって、情報パケットのビット数と余分に受け取ったビット数の比で表せるビットオーバーヘッド β を評価する必要がある。情報パケットの長さが ℓ で受け取った $\tilde{k}$ 個のパケットの長さが $\ell_1, \ell_2, \dots, \ell_{\tilde{k}}$ で表せるとき、 β は次式で計算できる。

$$\beta = \frac{\sum_{i=1}^{\tilde{k}} \ell_i - k\ell}{k\ell}$$

シフト分布が $(\Delta_0, \Delta_1, \dots, \Delta_{D_m})$ で与えられているとき、 $\ell_i \leq \ell + D_m$ であるので、 β は次の不等式を満足する。

$$\beta \leq \alpha + \frac{\tilde{k}D_m}{k\ell}$$

したがって、 m が定数で ℓ が十分に大きいときには、 β は α に収束する。

5.3.2 数値実験のパラメータ

この項の数値実験では、次のような設定を用いた。事前符号 C には (3, 30) 正則 LDPC 符号で $k = 900$, $n = 1000$ のものを用いている。パケット長は $\ell = 1000$ である。次数分布には文献(5)で与えられている $\Omega_1 = 0.007969$, $\Omega_2 = 0.493570$, $\Omega_3 = 0.166220$, $\Omega_4 = 0.072646$, $\Omega_5 = 0.082558$, $\Omega_8 = 0.056058$, $\Omega_9 = 0.037229$, $\Omega_{19} = 0.055590$, $\Omega_{65} = 0.025023$,

(注19)： G_b の作成の手順は 4.2.2 と同様である。

(a) パケットオーバーヘッド α と復号誤り率の比較

(b) ビットオーバーヘッド β と復号誤り率の比較

図 16 復号誤り率の比較 $D_m = 0$ は Raptor 符号の性能を表している。 $D_m = 1, 2, 3, 4, 5, 6$ は ZDF 符号の性能を表している。これらの図では曲線が左下に行くほど性能が良い。

$\Omega_{66} = 0.003135$ を用いた^(注20)^(注21) シフト分布には一様分布を用いた。すなわち、 $\Delta_0 = \Delta_1 = \dots = \Delta_{D_m} = 1/(1 + D_m)$ である。

5.3.3 復号誤り率の比較

図 16(a) ではパケットオーバーヘッド α と復号誤り率の比較をし、図 16(b) ではビットオーバーヘッド β と復号誤り率の比較をしている。 $D_m = 0$ は Raptor 符号の性能を表している。 $D_m = i$ はシフト量の最大値が i になっている ZDF 符号の性能を表している。復号誤り率とオーバーヘッドは小さいほど性能が良いので、これらの図では曲線が左下に行くほど性能が良い。図 16(a) 及び図 16(b) から ZDF 符号の方が Raptor 符号よりも復号性能が良いことが分かる。

6. おわりに

本稿では、ブロック消失訂正符号・LT 符号・Raptor 符号・シフトを利用したブロック消失訂正符号の符号化法並びに復号法を解説し、ZDF 符号の符号化法及び復号法と復号性能を紹介し

(注20)：この次数分布は Raptor 符号の最適化⁽⁵⁾ で与えられている次数分布であり、復号誤り率の低い Raptor 符号を与える。ZDF 符号では、別の次数分布を用いたほうが性能が良いことが見込まれるが、この次数分布を用いたとしても Raptor 符号よりも性能が良いことを 5.3.3 で数値実験的に明らかにする。

(注21)：表示のない Ω_i (例えば Ω_6) は全て 0 である。

た。本稿で説明した ZDF 符号はシフト操作という簡単な操作で符号化・復号が可能で良い復号性能を有していることを示した。

本稿では紹介しなかったが、文献(3)では数理的な手法で ZDF 符号の復号性能を明らかにしている。また、文献(22)では ZDF 符号に対する高速な復号法が与えられている。

謝辞 本稿を執筆する機会を与えて下さった情報理論研究専門委員会の皆様に感謝致します。特に、本稿の編集を担当して下さい東京理科大学の藤沢匡哉准教授に感謝します。また、匿名の読者の方々には初稿を丹念に御覧頂き、多くの有益なコメントを頂きました。この場をお借りしてお礼申し上げます。

文 献

- (1) J. Byers, M. Luby, and M. Mitzenmacher, "A digital fountain approach to asynchronous reliable multicast," IEEE J. Sel. Areas Commun., vol. 20, no. 8, pp. 1528–1540, 2002.
- (2) T. Nozaki, "Fountain codes based on zigzag decodable coding," Proc. Int. Symp. on Inf. Theory and its Applications (ISITA2014), pp. 274–278, Oct. 2014.
- (3) T. Nozaki, "Zigzag decodable fountain codes," IEICE Trans. Fundamentals, vol. 100, no. 8, pp. 1693–1704, Aug. 2017.
- (4) M. Luby, "LT codes," Proc. The 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS'2002), pp. 271–280, 2002.
- (5) A. Shokrollahi, "Raptor codes," IEEE Trans. Inf. Theory, vol. 52, no. 6, pp. 2551–2567, 2006.
- (6) R. Tanner, "A recursive approach to low complexity codes," IEEE Trans. Inf. Theory, vol. 27, no. 5, pp. 533–547, 1981.
- (7) N. Wiberg, H.-A. Loeliger, and R. Kotter, "Codes and iterative decoding on general graphs," Trans. Emerging Telecommunications Technologies, vol. 6, no. 5, pp. 513–525, 1995.
- (8) R.G. Gallager, "Low-density parity-check codes," IRE Transactions on Information Theory, vol. 8, no. 1, pp. 21–28, 1962.
- (9) M.G. Luby, M. Mitzenmacher, M.A. Shokrollahi, and D.A. Spielman, "Efficient erasure correcting codes," IEEE Trans. Inf. Theory, vol. 47, no. 2, pp. 569–584, 2001.
- (10) C.W. Sung and X. Gong, "A zigzag-decodable code with the MDS property for distributed storage systems," Proc. 2013 IEEE Int. Symp. Inf. Theory (ISIT), pp. 341–345, 2013.
- (11) M. Dai, C.W. Sung, H. Wang, X. Gong, and Z. Lu, "A new zigzag-decodable code with efficient repair in wireless distributed storage," IEEE Trans. Mob. Comput., vol. 16, no. 5, pp. 1218–1230, May 2017.
- (12) S. Gollakota and D. Katabi, "Zigzag decoding: combating hidden terminals in wireless networks," Proc. SIGCOMM, pp. 159–170, 2008.
- (13) A.A. Al-Shaikh and J. Ilow, "Packet loss recovery codes based on vandermonde matrices and shift operators," Proc. 2008 IEEE Int. Symp. Inf. Theory (ISIT), pp. 1058–1062, July 2008.
- (14) Y. Hanaki and T. Nozaki, "Erasure correcting codes by using shift operation and exclusive OR," Proc. Int. Symp. on Inf. Theory and its Applications (ISITA2018), pp. 281–285, 2018.
- (15) J. Qureshi, C.H. Foh, and J. Cai, "Optimal solution for the index coding problem using network coding over GF(2)," Proc. 9th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON), pp. 209–217, 2012.

- (16) X. Fu, Z. Xiao, and S. Yang, "Overhead-free in-place recovery and repair schemes of XOR-based regenerating codes," Proc. 2015 IEEE Int. Symp. Inf. Theory (ISIT), pp. 854–858, June 2015.
- (17) X. Gong, P. Hu, K.W. Shum, and C.W. Sung, "A zigzag-decodable ramp secret sharing scheme," IEEE Trans. Information Forensics and Security, vol. 13, no. 8, pp. 1906–1916, 2018.
- (18) S. Ogata and K. Ishibashi, "Zigzag decodable frameless ALOHA," Proc. 2018 52nd Asilomar Conference on Signals, Systems, and Computers, pp. 504–507, 2018.
- (19) S. Ogata and K. Ishibashi, "Application of zigzag decoding in frameless ALOHA," IEEE Access, vol. 7, pp. 39528–39538, 2019.
- (20) M. Oinaga, S. Ogata, and K. Ishibashi, "Zigzag decodable coded slotted ALOHA," Proc. 2018 15th Workshop on Positioning, Navigation and Communications (WPNC), pp. 1–6, 2018.
- (21) T. Emoto and T. Nozaki, "Shifted coded slotted ALOHA," Proc. Int. Symp. on Inf. Theory and its Applications (ISITA2018), pp. 291–295, Oct. 2018.
- (22) Y. Murayama and T. Nozaki, "Efficient scheduling of serial iterative decoding for zigzag decodable fountain codes," Proc. Int. Symp. on Inf. Theory and its Applications (ISITA2018), pp. 286–290, Oct. 2018.

(IT 研究会提案, 2019 年 3 月 18 日受付,
2019 年 4 月 23 日再受付)

野崎隆之 (正員)

平 20 東工大・工・情報卒. 平 22 同大学院修士課程了. 平 24 同大学院博士課程了. 博士 (工学). 同年日本学術振興会特別研究員. 平 25 神奈川大・助手. 平 27 山口大・助教. 平 29 同大学院創成科学研究科 (理学系) 講師. 符号理論の研究に従事. 平 23 SITA 奨励賞, 平 23 年度本会学術奨励賞, 平 24 SRC 論文賞, 平 26 IEEE ITJC Young Researcher Best

Paper Award, 平 29 年度本会論文賞各受賞.