Braided Convolutional Codes With Sliding Window Decoding

Kwame Ackah Bohulu

17-10-2017

1 Abstract

この論文には、編み畳み込み符号を復号するのに使用する目新しいスライディングウィンドウ復号アルゴリズムを紹介する。このアルゴリズムは BCJR アルゴリズムに基づいている。復号待ち時間と性能のトレードオフを調査し、復号の複雑さを減らすため、デコーディングウインドウに使用する均一と不均一なメッセージの受けまわしスケジュールと早期停止ルールを提案する。スライディングウィンドウ復号アルゴリズムのパラメータ、ウインドウの大きさとメッセージの受けまわしスケジュールをうまく選べるため、密度展開分析を行う。母符号より高いレートを得るため、定期的なパンクチュアリングを使用する。シミュレーション結果で明らかになたことは、不均一なメッセージの受けまわしスケジュールと定期的なパンクチュアリングを使用すると、幅広いレートで AWGN チャネル容量に近い性能、合理的な復号の複雑さと目が見えないエラーフローが可能です。

2 Introduction

編みブロック符号 (BBC) は二つの要素符号を相互接続し、接続状況は情報シンボルは両方の要素符号器でチェックして、お互いの要素符号のパリティーシンボルはお互いの入力になる。最近、[5] と [6] で紹介された BCH の要素符号の BBC と階段符号は、高速光通信に対する調査を行われ、繰り返し硬判定復号を使用することで性能が良いということがわかる。BBC に関する編み畳み込み符号 (BCC) は [7] で紹介され、BCJR アルゴリズムに基づく繰り返し復号が使用でき、ターボ符号と似たような符号です。しかしながら BCC は低長さ拘束長の畳み込み符号を要素符号とする。BCC の符号化方法それぞれの情報シンボルが二つの要素符号語が守られているような二次元スライドアレイで説明できる。要素符号の関係は、情報シンボルとパリティーシンボルが二次元スライドアレイに保存された位置で定義される。BBC に同質するしっかりと編んだ畳み込み符号 (TBC) は情報シンボルとパリティシンボルを保存するのに高密度配列を使用することによって作られている。一方, まばらに編んだ畳み込み符号 (SBC) は、低い高密度を持つので繰り返し復号化性能が改善された [7]。SBC 符号の最低距離は拘束長が大きくなるほど直線的に大きくなるということは [7] で数値的に示され、SBC は漸近的に良いと言われる。

この論文には、AWGN 通信路の上で送信する SBC 符号の復号化は [7] と [8] で使用する復号器の代わりにスライディングウインドウ復号器が提案される。スライディングウインドウ復号化は LDPC 畳み込み符号に関する研究がたくさん研究されており、復号化の性能、メモリー要件と待ち時間のトレードオフが簡単にできる。提案されたアルゴリズムと [7] と [8] の違いは BCJR アルゴリズムをしようする。

3 Sparsely Braided Convolutional Codes

SBC 符号は無限の二次元配列をしようして作られ、パリティーフィードバックで接続される二つの RSC 符号からなる。このように、情報シンボルとパリティービットシンボルが一緒編んでいる。SBC 復号は、bitwise と blockwise 二つのの種類にわかれている。bitwise は畳み込み符号インタリーバを使用し、送信が連続されている。blockwise はブロックインタリーバを使用し、有限長ブロックをそうしんする。この論文には、レート、$R = 1/3$ の blockwiseSBC 符号を使用する。システム図は図 1 に描かれていて、要素符号は、レート $R_{cc} = 2/3$ の RSC 符号からなる。情報系列は、大きさ T のブロックごとにわかれていて、$\mathbf{u} = (\mathbf{u}_0, \mathbf{u}_1, ..., \mathbf{u}_t, ...)$、$\mathbf{u}_t = (u_{t,1}, u_{t,2}, ..., u_{t,T})$。$P^{(0)}, P^{(1)}, P^{(2)}$ は長さ T のブロックインタリーバです。

$t = 0$ のとき、情報ブロック $\mathbf{u}_0$ と $\widetilde{\mathbf{u}}_0 = \mathbf{u}_0 P^{(0)}$ はそれぞれ要素符号 1 と要素符号 2 に一ビットずつ入力する。$\widetilde{\mathbf{v}}_{-1}^{(2)}$ と $\widetilde{\mathbf{v}}_{-1}^{(1)}$ は大きさ T のすべて 0 の系列で (初期条件)、それぞれ要素符号 1 と要素符号 2 に一ビットずつ入力する。それぞれの要素符号は長さ T のパリティーブロック $\hat{\mathbf{v}}_0^i = (\hat{v}_{0,1}^{(i)}, \hat{v}_{0,2}^{(i)}, ..., \hat{v}_{0,T}^{(i)}), i = 1, 2$ を出力する。$\hat{\mathbf{v}}_0^1, \hat{\mathbf{v}}_0^2$ と $\mathbf{u}_0$ を多重化し、通信路に送信する。

一般に、時間 t のとき、パリティーブロック $\hat{\mathbf{v}}_t^{(1)}$ は $\mathbf{u}_t$ と $\widetilde{\mathbf{v}}_t^{(2)} = \mathbf{v}_{t-1}^{(2)} P^{(2)}$ を使用して、要素符号 1 で計算する。パリティーブロック $\hat{\mathbf{v}}_t^{(2)}$ は $\widetilde{\mathbf{u}}_t = \mathbf{u}_t P^{(0)}$ と $\widetilde{\mathbf{v}}_t^{(1)} = \mathbf{v}_{t-1}^{(1)} P^{(1)}$ を使用して、要素符号 2 で計算する。$\hat{\mathbf{v}}_t^{(1)}, \hat{\mathbf{v}}_t^2$ と $\mathbf{u}_t$ 符号系列 $\mathbf{v} = (\mathbf{v}_0, \mathbf{v}_1, ..., \mathbf{v}_t ...)$ 多重化します。

$$\mathbf{v}_t = (v_{t,1}^{(0)}, v_{t,1}^{(1)}, v_{t,1}^{(2)}, v_{t,2}^{(1)}, v_{t,2}^{(0)}, v_{t,2}^{(1)}, v_{t,2}^{(2)}, ..., v_{t,T}^{(0)}, v_{t,T}^{(1)}, v_{t,T}^{(2)}) \tag{1}$$

この論文には、$R_{cc} = 2/3$ の RSC 符号では末尾ビットを使わないので最初と最後の状況は同じである。

4 Sliding Window Decoding

[7] には、平行パイプライン符号化が使用された。残念ながら、必要な復号化待ち時間が大きいである。[13]-[15] で紹介されたスライディングウインドウ復号化を使用することで、より小さい符号化待ち時間と大体同じ性能が可能です。ここで、blockwiseSBC 符号に関する目新しいの小さい待ち時間のスライディングウインドウ復号化方法を紹介する。

4.1 Window Decoding

AWGN 通信路で得た系列は $\mathbf{r} = (\mathbf{r}_0, \mathbf{r}_1, ..., \mathbf{r}_t ...)$。

$$\mathbf{r}_t = (r_{t,1}^{(0)}, r_{t,1}^{(1)}, r_{t,1}^{(2)}, r_{t,2}^{(1)}, r_{t,2}^{(0)}, r_{t,2}^{(1)}, r_{t,2}^{(2)}, ..., r_{t,T}^{(0)}, r_{t,T}^{(1)}, r_{t,T}^{(2)})$$

$\mathbf{l}^c = (\mathbf{l}_0^c, \mathbf{l}_1^c, ..., \mathbf{l}_t^c)$ は得られた通信路の LLR とする。

$$\mathbf{l}_t^c = (l_{t,1}^{c,(0)}, l_{t,1}^{c,(1)}, l_{t,1}^{c,(2)}, l_{t,2}^{c,(1)}, l_{t,2}^{(0)}, l_{t,2}^{c,(1)}, l_{t,2}^{c,(2)}, ..., l_{t,T}^{c,(0)}, l_{t,T}^{c,(1)}, l_{t,T}^{c,(2)})$$

$\mathbf{l}_t^c = (\mathbf{l}_t^{c,(0)}, \mathbf{l}_t^{c,(1)}, \mathbf{l}_t^{c,(2)})$ に分解することができる。$l_t^{c,j)}, j = 0, 1, 2$

4.2 Window Decoding Schedules

5 Density Evolution Analysis