Computing the Free Distance of Turbo Codes and Serially Concatenated Codes with Interleavers: Algorithms and Applications

Kwame Ackah Bohulu

2017/11/21

1 Algorithm For Computing the Free Distance of Turbo Codes and SCC Codes

ターボ符号と SCC 符号の自由距離を計算するのに使える二つのアルゴリズムを紹介する。このアルゴリズムを使うことで、符号の自由距離 (d_{free}), 自由距離をもつ符号語の数、(N_free) と d_{free} を作り出した情報系列の合計重み (w_free) が計算できる。

1.1 Algorithm to Compute the Free Distance of Turbo Codes

レート 1/3 のターボ符号があるとする。それぞれの要素符号のトレリスは Γ_1 と Γ_2 で示され、トレリスの長さは、(N+v) である。入力された情報系列は $\mathbf{u}^{(i)} = \{u_0, u_1, ..., u_{i-1}\}$ $i \leq N$ で示される。

アルゴリズムの核心 (かくしん) は、$\mathbf{u}^{(i)}$ と N-ビット情報系列 $\mathbf{u} = \{u_0, u_1, ..., u_{N-1}\}$ の最初の i ビットがひとし場合、$\mathbf{u}$ によって作られたターボ符号語の最小ハミング重みの計算である。

最小ハミング重み ($v(\mathbf{u}^{(i)})$) は、以下の式で計算する。

$$v(\mathbf{u}^{(i)}) = v_1(\mathbf{u}^{(i)}) + v_2(\mathbf{u}^{(i)})$$

$v_1(\mathbf{u}^{(i)})$ を計算する場合、$\mathbf{u}^{(i)}$ を要素符号 1 に入力して、符号のハミング距離を計算する。次に、状態 0 ($0_{\sum}$) に戻すのに必要 v ビットに対するパリティービットのハミング距離を計算する。

$$v_1(\mathbf{u}^{(i)}) = w_H(c_1^{(i)}) + v(\sigma_1^{(i)})$$

$v_2(\mathbf{u}^{(i)})$ を計算する場合、constrained subcode の考え方を使用する。入力シーケンス $\mathbf{u}^{(i)}$ は、基数 i の制約セット $V(\mathbf{u}^{(i)})$ を誘導し、$v_2(\mathbf{u}^{(i)})$ は制約付きサブコード $C_2(V(\mathbf{u}^{(i)}))$ の最小距離である。

$$v_2(\mathbf{u}^{(i)}) = min\, W_H({p_2}^{(i)})\ \ p_2 \in C_2(V(\mathbf{u}^{(i)}))$$

アルゴリズムの手順は以下で説明される。

【1.】$d_{free} = d*, N_free = 0, w_{free} = 0$. 二つの長さ 1 の情報系列、$\mathbf{u}^{(1)} = (1)$,$\mathbf{u}^{(1)} = (0)$ から始まる。

【2.a】$v(\mathbf{u}^{(i)}) = v_1(\mathbf{u}^{(i)}) + v_2(\mathbf{u}^{(i)})$ を計算する。$v(\mathbf{u}^{(i)}) \leq d_{free}$ の場合以外、$\mathbf{u}^{(i)}$ を捨てる。

【2b.】$\sigma_1^{(i)} = 0_{\sum}$ ($\sigma_1^{(i-1)} \neq 0_{\sum}$) の場合、$\mathbf{u} = \{\mathbf{u}^{(i)}, 0_{N-i}\}$ をターボ符号器に入力し、符号語 の重み $w_H(\mathbf{c})$ を計算する。もし $w_H(\mathbf{c})$ d_{free} であれば、N_{free} を 1 つ増やし、w_{free} を $w = w_H(\mathbf{u}^{(i)})$ 増やす。

もし $w_H(\mathbf{c}) < d_{free}$ であれば、d_{free} を $w_H(\mathbf{c})$ にし、N_{free} の値を 1 にし、w_{free} の値を w にする。

【3.】セット $\mathbf{u}^{(i)}$ に残したもの長さを 1 増やして、$\mathbf{u}^{(i+1)}$ を作って、前のステップをやる。

【4.】i=N の場合、セット $\mathbf{u}^{(i)}$ に残したものそれぞれターボ符号器に入力し、符号語を作る。もし $w_H(\mathbf{c})$ d_{free} であれば、N_{free} を 1 つ増やし、w_{free} を $w = w_H(\mathbf{u}^{(N)})$ 増やす。

もし $w_H(\mathbf{c}) < d_{free}$ であれば、d_{free} を $w_H(\mathbf{c})$ にし、N_{free} の値を 1 にし、w_{free} の値を w にする。

1.2 Algorithm to Compute the Free Distance of SCC Codes

レート 1/3 の SCC 符号があるとする。それぞれの要素符号のトレリスは Γ_1 と Γ_2 で示され、Γ_1 の長さは、(k+v) である。Γ_1 の長さは、(N+v) である。入力された情報系列は $\mathbf{u}^{(i)} = \{u_0, u_1, ..., u_{i-1}\}$ $i \leq N$ で示される。

アルゴリズムの核心 (かくしん) は、$\mathbf{u}^{(i)}$ と N-ビット情報系列 $\mathbf{u} = \{u_0, u_1, ..., u_{N-1}\}$ の最初の i ビットがひとし場合、$\mathbf{u}$ によって作られたターボ符号語の最小ハミング重みの計算である。

$v(\mathbf{u}^{(i)})$ を計算する場合、最初に $\mathbf{u}^{(i)}$ を要素符号 1 に入力して、符号語 $\mathbf{c}_1^{(i)}$ のハミング距離を計算する。次に、constrained subcode の考え方を使用する。$\mathbf{c}_1^{(i)}$ は、基数 (きすう)2i の制約セット $V(\mathbf{c}_1^{(i)})$ を誘導し (ゆうどうし)、$v(\mathbf{u}^{(i)})$ は制約付きサブコード $C_2(V(\mathbf{u}^{(i)}))$ の最小距離である。

$$v(\mathbf{u}^{(i)}) = min\, W_H({c_2}^{(i)})\ \ c_2 \in C_2(V(\mathbf{u}^{(i)}))$$

アルゴリズムの手順は以下で説明される。

【1.】$d_{free} = d*, N_free = 0, w_{free} = 0.$ 二つの長さ 1 の情報系列、$\mathbf{u}^{(1)} = (1)$,$\mathbf{u}^{(1)} = (0)$ から始まる。

【2.a】$v(\mathbf{u}^{(i)}) = v_1(\mathbf{u}^{(i)}) + v_2(\mathbf{u}^{(i)})$ を計算する。$v(\mathbf{u}^{(i)}) \leq d_{free}$ の場合以外、$\mathbf{u}^{(i)}$ を捨てる。

【2b.】$\sigma_1^{(i)} = 0_{\sum}$ ($\sigma_1^{(i-1)} \neq 0_{\sum}$) の場合、$\mathbf{u} = \{\mathbf{u}^{(i)}, 0_{N-i}\}$ を SCC 符号器に入力し、符号語 $\mathbf{c}$ の重み $w_H(\mathbf{c})$ を計算する。もし $w_H(\mathbf{c})$ d_{free} であれば、N_{free} を 1 つ増やし、w_{free} を $w = w_H(\mathbf{u}^{(i)})$ 増やす。

もし $w_H(\mathbf{c}) < d_{free}$ であれば、d_{free} を $w_H(\mathbf{c})$ にし、N_{free} の値を 1 にし、w_{free} の値を w にする。

【3.】セット $\mathbf{u}^{(i)}$ に残したもの長さを 1 増やして、$\mathbf{u}^{(i+1)}$ を作って、前のステップをやる。

【4.】i=N の場合、セット $\mathbf{u}^{(i)}$ に残したものそれぞれターボ符号器に入力し、符号語を作る。もし $w_H(\mathbf{c})$ d_{free} であれば、N_{free} を 1 つ増やし、w_{free} を $w = w_H(\mathbf{u}^{(N)})$ 増やす。

もし $w_H(\mathbf{c}) < d_{free}$ であれば、d_{free} を $w_H(\mathbf{c})$ にし、N_{free} の値を 1 にし、w_{free} の値を w にする。