整数リング上において順列多項式を使用するターボ符号のためのインタリーバ

Abstract-この論文には $\mathbb{Z}_n$ の上において順列多項式に基づいて決定論インタリーバを紹介します。このインタリーバの主な特徴は与えた部品コードによって代数的に設計することができます。その上, インタリーバテーブルを保存する必要がありません。順列多項式に基づいてインタリーバを使うと、インタリーバの設計は多項式の係数を選ぶことになります。順列多項式に基づいてインタリーバで作られた TC の性能は普通に入力重み 2 エラーイベントの集合に影響を与えるということが観察されました。この集合の最低距離と多重度を設計条件として、良い順列多項式を選ぶために使われています。m が小さい場合、そのエラーイベントを分析する簡単な方法も紹介されました。良いインタリーバの探しが行われました。長いフレームサイズの場合は紹介されたインタリーバの性能は S-ランダムインタリーバの性能とだいたい同じです。短いフレームサイズの場合は S-ランダムインタリーバよりよい働きをします。
ケーワード：整数リング、インタリーバ、順列多項式、ターボ符号、重みスペクトラム

1 Introduction

ターボ符号（ＴＣ）は二つの畳み込み符号をインタリーバにより、並列連結をして作られています [1]。良い性能のＴＣを作るため、特に短いフレームサイズの場合において、インタリーバの設計は大変重要です。そのため、ＴＣを作るためのインタリーバはたくさん研究されており、一般的にランダムインタリーバと決定論インタリーバの二つのグループにわかれています [2]-[6]。

1.1 ランダムインタリーバ

1.1.1 単純なインタリーバ

最も単純ランダムインタリーバは情報ビットを疑似的に並びかえることです。長いフレームサイズの場合はシャンオンリミットに近づけることが可能ということが論文「１」で示されました。

1.1.2 Ｓ－ランダムインタリーバ

Ｓ－ランダムインタリーバは単純なランダムインタリーバを改善したもです。距離Ｓ以内の入力した二つのポジションは距離Ｓ以内の出力ポジションに並び替えてはいけないという規制があります。その規制は、一つの部品コードにある短いエラーエベントがもう一つの部品コードにある短いエラーエベントをマッピングすることを防止します。

Ｓ－ランダムインタリーバよりよい性能を持つランダムインタリーバは論文「３」、「４」で発表されています。ターボ符号のためのランダムインタリーバは暗号器と復号器の両方にインタリーバテーブルを保存しなければなりません。大きいフレームサイズの場合には望ましくないです [7] 。

1.2 決定論インタリーバ

決定論インタリーバはインタリーバテーブルを使わずに、アルゴリズムでインタリーブとデインタリーブが可能です。最も単純な決定論インタリーバはブロックインタリーバ [8] とリネアーインタリーバ [6] です。ある部品コードがあり、短いフレームサイズの場合は、インタリーバのパラメータをうまく選べばランダムインタリーバより良い働きをします。しかし、長いフレームサイズの場合は高いエラーフローがあるので、ランダムインタリーバのほうがいいと言われています。

1.2.1 二次インタリーバ

二次インタリーバは論文「６」で紹介されており、二次合同に基づいて決定論インタリーバです。$N = 2^n$ の場合

$$c_i = \frac{(ki(i+1))}{2}(modN)$$

は k が奇数ならば、i $\in \{0, 1, ..., N-1\}$ の順列であることが証明できる。すると二次インタリーバはすべての i において以下のように表されます。

$$\Pi_{\varrho N} : c_i \mapsto c_{(i+1)modN}$$

二次インタリーバはランダムインタリーバの平均的な性能が可能です [6]。この論文では整数リングの上順列多項式に基づいて新しい決定論インタリーバを提案しました。多項式の係数をうまく選べばＳ－ランダムインタリーバよりよい性能をもつ可能があるということを主張しました。

2 $\mathbb{Z}_n$ の上において順列多項式

a_0, a_1, a_2, a_m が非負の整数のとき、ある多項式 $P(x) = a_0 + a_1x + a_2x^2 + ... + a_mx^m$ に $N \geq 2$ の整数 N を与え、P(x) が $\{0, 1, 2, ..., N-1\}$ を並び替えることを $\mathbb{Z}_n$ における多項式と呼ぶ。$P(x)$ のフォーマール誘導体は以下のように定義します。

$$P'(x) = a_1 + 2a_2x + 3a_3x^2 + ... + ma_mx^{m-1} \quad (1)$$

注意：この論文にある掛け算と足し算は明示的に書かれていなかったため、モヅロNで計算しました。ある多項式を順列多項式と呼ぶ必要十分条件は、整数Nによってちがいます。

2.1 $N = 2^n$

定理 2.1 $P(x) = a_0 + a_1x + a_2x^2 + ... + a_mx^m$ は整数係数多項式である。
1)a_1 は奇数で
2)$a_2 + a_4 + a_6 + ...$ は偶数で
3) $a_3 + a_5 + a_7 + ...$ は偶数
である場合にのみ、P(x) は整数リング $\mathbb{Z}_{2^n}$ の上において順列多項式である。

2.2 $N = p^n$,p は素数である

定理 2.2: P(x) は $\mathbb{Z}(P)$ の上において順列多項式であり、P'(x) は $\mathbb{Z}_{p^n}$ のすべての整数 x を入れて、0 modulo p にならない場合にのみ、P(x) は整数リング $\mathbb{Z}_{p^n}$ の上において順列多項式である。

2.3 $N = \sum_{i=1}^{m} {p_i}^{n_{pi}}$

定理 2.3: P(x) は順列多項式 $mod{p_i}^{n_p i}$ すべての i である場合にのみ, P(x) は順列多項式 mod N である。

2.4 二次順列多項式の条件

推論 2.4:$P(x) = ax + bx^2$ の二次多項式は a 0 で、b=0 mod p 場合にのみ、P(x) は整数リング $\mathbb{Z}_{p^n}$ の上順列多項式である

2.5 順列多項式に関する定義

2.5.1 定義 2.6 (ベース)

$N = \sum_{i=1}^{m} {p_i}^{n_{pi}}$ の場合、N のベースはベクトル $p_N = [p_1, p_2, p_3, ..., p_m]$ のように定義します。

2.5.2 定義 2.7 (N のオーダー)

$N = \sum_{i=1}^{m} {p_i}^{n_{pi}}$ の場合、N のオーダーはベクトル $o_N = [n_{p_1}, n_{p_2}, n_{p_3}, ..., n_{p_m}]$

N に対して、与えられたら x は、以下のように書くことができます。
$x = x_0 \sum_{i=1}^{m} pi^{o_x[i]}$, $p_i \in p_N$ x_0 と p_i は互いに素である。

2.5.3 定義 2.8 (N に対する x のオーダー)

N に対する x のオーダーは、ベクトル $o_x = [o_x[1], o_x[2], o_x[3], ..., o_x[m]]$

2.5.4 数のオーダーの性質

性質 2.9: $z = xy$ ならば、$o_z = o_x + o_y$ である
性質 2.10: もし x が y を分割できれば、$z = x/y$ とすると $o_z = o_x + o_y$ である
性質 2.11: $z = x^y$ とすると、$o_z = yo_x$ である

3 順列多項式に基づいたインタリーバ

インタリーバの働きは数を並び替えることです。順列多項式も同じ働きをします。例：N=8 の場合、多項式 $P(X) = 2X^2 + X + 3$ は定理 2.1 の条件を満たしているので、$\mathbb{Z}_N$ の上において順列多項式である。整数 x を $P(X)$ で並び替えたら、順序 {0,1,2,3,4,5,6,7} は {3,6,5,0,7,2,1,4} に並び替える。一般的に、多項式 $P(X)$ は $\mathbb{Z}_N$ の上において順列多項式であったら、順列多項式に基づいてインタリーバは以下のように定義します。

$$\Pi_{\mathcal{P}_\mathcal{N}} : x \mapsto P(x)$$

多くのインタリーバの性能は入力重み 2 と言うエラーイベントに影響を与える [14]。入力重み 2 のエラーイベントと言うのは、二つの情報ビットが間違っているというエラーイベントのことです。ターボ符号の入力重み 2 エラーイベントはそれぞれの部品コードにある一つの入力重み 2
エラーイベントに対応します一番目の部品コードに起こる入力重み 2 エラーエベントは $(x, x+t)$ のように代表します。(このペアは入力重み 2 エラーエベントの位置を示す。) その二つのポジションは二番目の部品コードの $\pi(x)$ と $\pi(x+t)$ に並び替える。$\pi(x)$ と $\pi(x+t)$ の間の距離 $\Delta(x,t)$ は以下のように表せられる。

$$\Delta(x,t) = \pi(x+t) - \pi(x) mod N$$

$\pi(x)$ と $\pi(x+t)$ がかなり近い点であることに興味があります。それぞれのインタリーバのパフォーマンスを比べる（くらべる）ために Δ(x,t)=0 の線に近い点を見ます。$\Delta(x,t) = 0$ の線図の上と下にあります。

4 順列多項式の探し方

順列多項式に基づいてインタリーバの場合は、良い係数をうまく選べば $\Delta(x,t) = 0$ の線にかなり近い点がなくなります。順列多項式に基づいてインタリーバはそれぞれのパフォーマンスが違います。フレームサイズと部品コードが与えられたら、一番良い順列多項式を見つけようとするのである。固定フレームサイズが与えたら、残りの変数は多項式の次数と係数である。この論文では以下のようの二次多項式に注目します。

$$P(x) = bx\ + ax + c$$

二次多項式は可能な限り低い複雑さをもっており、分析が相対的に簡単からです。定数項 c はインタリーブした順序にある循環回転しか対応しません。

順列多項式の条件と関係ないので、0 とし、以下のような多項式を注目します。

$$P(x) = bx \ + ax$$

多項式の良い係数を選ぶために、ターボ符号のエラーイベントの部分集合の最低距離を基準とします。その部分集合は入力重み 2m エラーイベントです。そのエラーイベントは簡単に見つけて、数えるからです。この論文にはそれぞれの部品コードが (tail-biting trellis) を使っている仮定を使います。その仮定を使うと、終了によって境界効果を無視することができます。

4.1 入力重み $2m$ エラーイベント

順列多項式はリニアインタリーバの一般化のように考えることができます。リニアインタリーバと同じように、順列多項式によく起きるエラーイベントは入力重み 2 mエラーイベントです。$m = 1, 2, ...$ しかし、多項式の係数の a と b をうまく選べばそのようなエラーイベントを制御することができます。部品コードが与えたら、良い性能をもつ二つの係数が見つけられます。代表的な入力重み m エラーイベントは図 3 に現れます。それぞれの入力重み 2 エラーイベントは最初と最後のポジションを示す整数の組で代表できます。二つの部品コードは同じ組織畳み込み符号を使用するので、すべての t_i と s_i は畳み込み符号の (cycle length)τ の複数です。この論文では、(cycle length)τ というのは入力順序が [1,0,0,0.....] とき、符号器の出力の周期である。
例
(cycle length) ＝最低入力重み 2 エラーイベントの距離　1 。エラーパタンは以下の長さ 2 mのベクトルのように定義します。$[t_1, t_2, ..., t_m, s_1, s_2, .., s_m]$
図 3 入力重み 2 エラーイベントで、以下のm式が書けます。

$$P(x_2) - P(x_1) = s_1 \tag{3.1}$$

$$P(x_3) - P(x_1 + t_1) = s_2 \tag{3.2}$$

$$P(x_4) - P(x_2 + t_2) = s_3 \tag{3.3}$$

$$P(x_m + t_m) - P(x_{m-1} + t_{m-1}) = s_m \tag{3.m}$$

t_i,s_i は τ の小さい複数,x_i $[0, 1, ...N - 1]$
エラーイベントの見つける方法を簡単にするために、式 3 を分析しやすい形に変換します。すると以下のようになります。

$$\begin{aligned} s_1 - s_2 + s_3 - s_4... = {} & 2b(x_1t_1 - x_2t_2 + x_3t_3 - x_4t_4...) \\ & + b((t_1)^2 - (t_2)^2 + (t_3)^2 - (t_4)^2...) \\ & + a(t_1 - t_2 + t_3 - t_4...) \end{aligned} \tag{4}$$

$$\sum_{i=1}^{m}(-1)^{i-1}s_i = 2b\sum_{i=1}^{m}(-1)^{i-1}x_i t_i + b\sum_{i=1}^{m}(-1)^{i-1}{t_i}^2 + a\sum_{i=1}^{m}(-1)^{i-1}t_i \tag{5}$$

図 3 のような入力重み 2 mエラーイベントが現れるために、式 5 は式 3 の残りのm－ 1 式と一緒に使わなければなりません。この論文には、入力重み 2 mエラーイベントを見つけるために式 3 ～ 5 を使います。あるエラーパターンが与えられ、重ならなかったら、エラーイベントのハミング距離を一意に決定できます。例で説明します。部品コードは 5/7 の RSC コードとします。そのコードの $\tau = 3$。t_i と s_i は τ の複数なので kτ の一般的な形を持っています。入力は、$1 + D^{k\tau}$ とします。出力は以下のようになります。

$$\begin{aligned}(1+D^{3k})\frac{1+D^2}{1+D+D^2} &= (1+D^3+D^(2.3)+...+D^{3(k-1)}) \\ &\times (1+D^3)\frac{1+D^2}{1+D+D^2} \\ &= (1+D^3+D^(2.3)+...+D^{3(k-1)}) \\ &\times (1+D+D^2+D^3)\end{aligned} \tag{6}$$

入力の $1 + D^3$ にたいして出力順序の重みは $w_0 = 2$ になります。（最初と最後の 1 を入れずに）そのうえ入力の $1 + D^3k$ にたいして出力順序の重みは $2 + w_0k$ になります。エラーエベントの全出力重みは以下のようになります。

$$6m + \left(\frac{\sum|t_i|}{\tau} + \frac{\sum|s_i|}{\tau}\right)w_0 \tag{7}$$

この論文では式 (7) を使ってエラーイベントのハミング距離を計算します。

4.2 効果的な自由距離 (d_{eff}) を使用して、良いインタリーバを探します。

決定論インタリーバでは大きな d_{eff} 良い性能を表すわけではないですが、小さい d_{eff} だとほとんど悪い性能になる関係があります。このように悪い順列多項式を選ばないように、d_eff を基準とします。順列多項式に基づいてインタリーバを使う場合、多項式の係数をうまく選べば、ある部品コードによく起きる重み 2 エラーイベントが防止できます。そうすると、それより大きい入力重み 2 エラーイベントを分けることができます。1 番目の部品コードに起きる入力重み 2 エラーイベントの長さを t+1 とします。そうすると t は τ の複数で、t のオーダーは o_t とします。すると、2 番目の部品コードに起きる入力重み 2 エラーイベントの長さ引く 1 は以下のようになります。

$$\Delta(x,t)\ P(x+t) - P(x) = 2btx + bt^2 + at = c_1x + bt^2 + at \tag{8}$$

性質 2.9 で $o_{c1} = o_2 + o_b + o_t$ x に従って c_1x の図を描くと $p_N^{(on-oc1)}$ 水平線が出ます。$bt^2 + at$ は水平線のオフセットを与えます。短い入力重み 2 エラーイベントを防止するために、 t が τ の小さい複数の場合 $\Delta(x,t)$ も τ の複数の値を 0 から離れてほしい。これをするため、ベクトル o_{c1} を大きくしたい。o_{c1} はもう大きいので、最初の線しか興味ありません。 0 からの距離は以下のように書けます。

$$s = \pm\Delta(x,t) mod p_N^{o_{c1}} = (bt^2 + at) mod p_N^{o_{c1}} \tag{9}$$

a,b,τ が与えたら、$\mathbf{L}_{(a,b,\tau)}$ は以下のように定義して、良いインタリーバを選ぶ基準とします。

$$\mathbf{L}_{(a,b,\tau)}\ min\ (|s| + |t|)$$

ある部品コードが与えたら、$\mathbf{L}_(a,b,\tau)$ から d_{eff} を選ぶことができます。良い a と b をさがすとき、範囲を制限したらよいです。以下の補題で a と b の班を制限することができます。

補題 4.1 入力重み 2 エラーイベントの分析では、b を $b_1 \cdot b_0$=$b_1 \cdot p_N^{o_{b1}}$ ようにかけば b1 を 1 とすることができます。

補題 4.2 入力重み 2 エラーイベントの分析では、$b = b_1 \cdot p_N^{o_{b1}}$ が与えたら、a は $1 \leq a \leq p_N^{o_{b1}}$ だけ使えば十分です。7/5 部品コードの場合の結果がテーブル 1 に書いてあります。

o_b が与えられたら、d_{eff} で良い a と b を選ぶのは十分可能のようです。しかし、d_{eff} は o_b を選ぶ十分情報ではありません。例えば式 (9) を見ると、o_b が大きければもっと良い a を選ぶことができます。でもシミュレーションでわかるのが、o_b はある値まで順列多項式の性能が良くなって、その値を超えると性能が悪くなります。それを説明して、これより正確パラメータを選ぶ方法を見つけるために、もっと高い入力重みエラーイベントを調べなければなりません。

4.3 もっと高い入力重みエラーイベント

良い順列多項式を探すとき、入力重み 2 エラーイベントの d_{eff} で決めます。d_{eff} を見つけるために、m が小さい値しか注目しません。大きい値はほとんど大きいハミング距離と関係があるからです。エラーイベントを見つける一つの方法はエラーパタン $[t_1, ..., t_m, s_1, ..., s_m]$ をきめて x_1 を計算します。
エラーパターンと x_1 が決めたら、残りの x_i は、式（３）の残りの $m-1$ 式で計算できます。最後に x_i, t_i, s_i の $3m$ 値をまだ使っていない（３）か（４）式に使って、エラーイベントが正しいかどうかを確かめます。

順列多項式に基づいてインタリーバは高度に構造化してるので　を 0 から $p_N^{(o_N-o_2-o_b)}-1$ から確認したら十分です。$p_N^{(o_N-o_2-o_b)}$ が小さい場合、この方法は有能である。エラーパターンが与えられ、$_1$ を見つけるとき、入力重み 2 mエラーイベントの制約を一つの式に変えたいです。つまり、式（３）での $m-1$ 残りの式を式（５）で $i=2,...m$ をキャンセルします。以下の問題を解決できれば、（３）と（５）を一つの式にすることができます。
問題：N,a,b,s が与え、$P(y)-P(x)=s$ なら、x を y の関数とします。
以上の問題を解決する前、ほかの定義が必要。順列多項式 $P(x)=bx^2+ax$ と定数した s が与えたら、順序 $\{y_i\}$ は以下のように定義します。

$$\begin{aligned} P(y_0)-P(0)&=s\\ P(y_1)-P(1)&=s\\ P(y_2)-P(2)&=s\\ P(y_3)-P(3)&=s \end{aligned} \tag{16}$$

そして、$\Delta_k(i)$ を再帰的に以下のように定義します。

$$\begin{aligned} \Delta_1(i)&=y_i-1\\ \Delta_2(i)&=\Delta_0(i+1)-\Delta_0(i)etc \end{aligned}$$

注意：y_i と Δ_k (i) は a,b と s に関する関数です。
すると以下の定理が出ます。

定理 4.3 N,P(x) と s が与えたら、すべての i で $\Delta(i)$ は同じオーダを持っています。$\Delta(i)$ のオーダーは o_{Δ_k} と示したら、k>0 場合 $o_{\Delta_0} = o_s$。そして、$o_{\Delta_k} = o_{\Delta_{k-1}} + o_b + o_{2k}$。すべての k では $o_{\Delta_k} = ko_b + ko_2 + \sum_{n=1}^{k} o_n + o_s$ である。推論 4.4 N,P(x) と s が与え、K が一番大きい数で、$ok \not\succeq o_N$ の場合、すべての i で $\Delta_K(i) = \Delta_K$ は定数である。注意：K, N, a, b と s の関数である。

推論 4.4 での K を見つけられたら、すべての k>K 場合 $\Delta_K(i) = 0$。その上、もしすべての $0 \leq k \leq$ K で,$\Delta_K(0)$ が基地であるなら、　すべての i で $\Delta_K(i)$ の定義で $\Delta_K(i)$ を計算すくことができます。わかりやすくするために、Δ_k を $\Delta_k(0)$ と代表します。s を指す必要があったら、$\Delta_k(s)$ を使います。
これで、$P(y) - P(x) = s$ の場合、x と y の関係を見つける道具を持っています。以下の定理で集約しました。

定理 4.5 N,P(x) と s が与え、$P(y) - P(x) = s$ であるならば

$$y = x + \Delta_0(s) + \Delta_1(s) + \frac{(x(x-1)}{2!}\Delta_2(s) + \frac{(x(x-1)(x-2)}{3!}\Delta_3(s) + \qquad (17)$$

x<k の場合 $\binom{x}{k} = 0$ と定義したら、17 が以下のように書けます。

$$y = F(x,s) \triangleq x + \sum_{k=0}^{\infty} \binom{x}{k} \Delta_k(s) \qquad (18)$$

F(x,s) を整数値の多項式をするために、N が与えたら、$D(k) = \prod_{i=2}^{k} \frac{i}{p_N^{o_i}}$ を定義します。$\frac{\Delta_k(s)}{k!}$ はいつも整数であるわけではないからです。

4.4　式を解くことで、エラーイベントを探す。

式 (4) が正解であったら、エラーイベントを作ることができます。定理 4.5 を式 (3) の残り $m-1$ で使えば、以下の式が出ます。

$$\begin{aligned} x_2 &= F(x_1, s_1) \\ x_3 &= F(x_1 + t_1, s_2) \\ x_4 &= F(x_2 + t_2, s_3) \\ x_m &= F(x_{m-2} + t_{m-2}, s_{m-1}) \end{aligned} \qquad (20)$$

そして、(20) を (4) で使えば、一般的な x_1 に対する多項式になります。　一般の多項式では、解答、またはいくつの解答があるかを探すことの複雑さは高いです。しかし,m の値が小さい場合もっと簡単な方法があります。これから、$m = 1, 2, 3$ の場合を解決します。

4.4.1 $m = 1$、入力重み 2 エラーイベント

以上の場合は、式 (20) を使わずに、式（ 4 ）は以下のようになります。

$$2bt_1x_1 + bt_1{}^2 + at_1 - s_1 = 0 \tag{21}$$

x_1 に対する線形多項式としたら、以下のようになります。

$$c_1x + c_0 \tag{22}$$

もし $o_{c0} \geq o_{c1}$ 場合のみ (22) の解答があります。その条件が満たされる場合、$p_N^{o_{c1}}$ で (22) を分割でき、結果は特別な解答を持つ一次多項式 $\bmod p_N^{o_N - o_{c1}}$ になります。多くの場合ではエラーイベントの位置よりエラーイベントのハミング距離とそのハミング距離の多重度に興味があります。ですから、式 (22) を解答するかわりに $o_{c0} \geq o_{c1}$ を確認する。満たされていたら、$p_N^{o_{c1}}$ の多重度を対応するスペクトラム線と足します。

4.4.2 $m = 2$、入力重み **4** エラーイベント

(20) 使って、(4) は以下のようになります。

$$2b[\, x_1t_1 - (x_1 \sum_{k=0}^{\infty} \binom{x_1}{k} \Delta_k(s_1))t_2]\, b(t_1^2 - t_1^2) + a(t_1 - t_2) - (s_1 - s_2) = 0 \tag{23}$$

x_1 に関する条項を収集すると以下のようになります。

$$\begin{aligned} &(-2b\Delta_0(s_1)t_2 + b(t_1^2 - t_2^2) + a(t_1 - t_2) - (s_1 - s_2)) \\ &\quad + \ 2b(t_1 - t_2 - \Delta_1(s_1)t_2)x_1 \\ &\quad - 2bt_2 \sum_{k=2}^{\infty} \frac{\Delta_k(s_1)}{k!} \prod_{m=0}^{k-1} (x_1 - m) = 0 \end{aligned} \tag{24}$$

分数係数を持つような x_1 に対する多項式です。s_1 が与えられ、当然の結果 4.4 で K を見つけることができます。(24) を D(K) と掛けたら、以下のようになります。

$$c_0 + c_1x_1 + c_2(x_1) = 0 \tag{25}$$

$$\begin{aligned} c_0 &= (-2b\Delta_0(s_1)t_2 + b(t_1^2 - t_2^2) + a(t_1 - t_2) - (s_1 - s_2))D(K) \\ c_1 &= 2b(t_1 - t_2 - \Delta_1(s_1)t_2)D(K) \\ c_2(x_1) &= -2bt_2 \sum_{k=2}^{\infty} \frac{\Delta_k(s_1)}{k!} \prod_{m=0}^{k-1} (x_1 - m) = 0. \end{aligned} \tag{26}$$

c_0 と c_1 は整数であり、$c_2(x_1)$ は x_1 に対する多項式です。(26) のオーダーは少なくとも、$3o_b + 3o_2 + o_s1 + o_{t2}$。当然の結果で、$\frac{c_1x_1+c_2(x_1)}{(p_N^{o_{c1}}}$ は順列多項式である。$t_1 \neq t_2$ 場合、推論 2.5 を使うために、以下の条件あります。

$$o_{c1} \ll 3o_b + 3o_2 + o_{s1} + o_{t2} \tag{27}$$

エラーパターン $[t_1, t_2, s_1, s_2]$ が与えたら、(27) の条件があっているかどうかを確認します。そして、入力重み 2 エラーイベントと同じように $o_{c0} \geq o_{c1}$ の確認しかしません。正しければ、$p_N^{o_{c1}}$ 同じエラーパタンを持っているエラーイベントがあります。もしスペクトラムしか興味がなかったら、$\Delta_0(s_1)$ と $\Delta_1(s_1)$ を解決することになります。

4.4.3 $m = 3$、入力重み **6** エラーイベント

この場合はの (25) は入力重み 4 エラーイベントと同じ形になります。

$$c_1 = 2b(t_1 - t_2 + t_3 - \Delta_1(s_1)t_2 + \Delta_1(s_2)t_3)DD = D(max(K(s_1), K(s_2)))$$

そして、$c_2(x_1)$ の係数のオーダーは少なくとも $min(3o_b + 3o_2 + o_{s_1} + o_{t_1}, 3o_b + 3o_2 + o_{s_2} + o_{t_3})$。推論 2.5 を使用する条件は

$$o_{c1} = min(3o_b + 3o_2 + o_{s_1} + o_{t_1}, 3o_b + 3o_2 + o_{s_2} + o_{t_3}) \tag{28}$$

あっていたら、$\frac{c_1x_1+c_2(x_1)}{p_N^{o_{c1}}}$ は順列多項式である。もしスペクトラムしか興味がなかったら、$\Delta_0(s_1), \Delta_0(s_2), \Delta_1(s_1)$ と $\Delta_1(s_2)$ を解決することになります。

4.5 o_b の上界

定理 2.5 で o_b の上の上界を見つけることができます。

4.5.1 入力重み 4 エラーイベントでの o_b の上界

(25) から始まります。条件 (27) があっていて、$o_{c0} \geq o_{c1}$ のとき、(25) で x_1 の解答があるなら、与えたエラーパターンにたいして x_1 から始まるエラーイベントがあります。0 から N-1 のすべて x_1 が解答である特別な場合があります。$t_1 = t_2 = t, s_1 = s_2 = s$ のとき、(26) にある c_0 と c_1 は以下のようになります。

$$c_0 = -2b\Delta_0(s)tD(K)$$
$$c_1 = -2b\Delta_1(s)tD(K)$$

$o_{c0} \geq o_{c1}$ と言うことが簡単に見えます。$o_{c0} = o_N$ のとき、(25) は全 0 の多項式になります。

4.5.2 入力重み 6 エラーイベントでの o_b の上界

(25) から始まります。条件 (28) があっていたら、c_0 と c_1 は以下のようになります。

$$\begin{aligned} c_0 =&[\,2b[\,t_3\Delta_0(s_2) - t_2\Delta_0(s_1)] + 2bt_3t_1\Delta_1(s_2) \\ &+ 2bt_1t_3 + b(t_1^2 - t_2^2 - t_3^2) \\ &+ a(t_1 - t_2 + t_3) - (s_1 - s_2 + s_3)]\,D \end{aligned} \tag{30}$$

$$c_1 = [\,2b(t_1 - t_2 + t_3) + 2b(t_3\Delta_(s_2) - t_2\Delta(s_1))] \tag{31}$$

$D = D(max\ (k(s_1), k(s_2)))$ で $c_2(x_1)$ の係数のオーダーは o_c1 より大きいすべての　がエラーパタンの解答である場合に興味あります。
[2t,t,-t,s,-s,2s] の形を持つエラーパタンが大変重要です。最小ハミング距離に対応するエラーパタンは $t = \tau$ と $s = \pm t$ のときです。以上のエラーパタンで $t_1 - t_2 + t_3 = s_1 - s_2 + s_3 = 0$ それで、c_0 と c_1 は以下のようになります。

$$c_0 = \left[\,-2bt[\,\Delta_0(-s) + \Delta_0(s)] - 4bt^2\Delta_1(-s)\right]\,D \tag{32}$$

$$c_1 = -2bt(\Delta_1(-s) + \Delta_1(s))D \tag{33}$$

ここから進めるために二つの当然の結果が必要です。補題 4.6 $\Delta_0(-s)+\Delta_0(s)$ のオーダーは $o_b + o_2 + 2o_b$ 補題 4.7 $\Delta_1(-s)+\Delta_1(s)$ のオーダーは少なくとも $o_b + 2o_2 + 2o_b$ それで c_0 と c_1 のオーダーを計算することができます。

$$o_{c0} \geq 2o_b + 2o_2 + 3o_\tau$$

$$o_{c1} \geq 2o_b + 3o_2 + 3o_\tau$$

ベクトル o_N にあるそれぞれのメンバー $2o_b + 2o_2 + 3o_\tau$ に対応するメンバーより大きくないとき c_0 と c_1 は $0 mod N$ になって、すべての x_1 は式の解答である。o_b の上界とすることができます。

4.6 a と b を探すときの範囲

補題 4.1 と補題 4.2 で、入力重み 2 エラーイベントのとき、o_b が決めたら $b = b_0 \cdot p_N^{o_b}$ として a は 1 から b_0 しか注目しない。残念ながら一般的な場合でも a の範囲の結果はだいたい同じです。

定理 4.8 二次順列多項式に基づいたインタリーバで、a の範囲は 1 から 2b を注目しなければなりません。

5 結果

フレームサイズ N と部品コードが与えられたら、良い順列多項式に基づいてインタリーバを探すことは、多項式の a と b を計算することになります。最初に、o_b の値を決めます。前の分析で $p_N^{o_b}$ を大きくしなければならないです

が、特別入力重み 4 エラーイベントと入力重み 6 エラーイベントで成約を拘束しなければなりません。o_b が決めたら、$b = p_N^{o_b}$ として定理 4.8 の範囲ですべての a を計算する。