幅 3 の共同疎形式

桶屋 勝幸

(株) 日立製作所 システム開発研究所,
〒 244-0817 横浜市戸塚区吉田町 292 番地

katsuyuki.okeya.ue@hitachi.com

あらまし 楕円曲線暗号におけるマルチスカラー倍算の高速化手法として、スカラー対を共同疎形式 (JSF) を用いて表現する方法がある。Solinas の JSF 提案論文では、三つの拡張形式の構成が未解決問題として提示された。そのうちの二つは解決済みであるが、残りの一つが未解決として残されていた。その未解決問題とは、通常の JSF が $0, \pm 1$ によって定まる桁集合を用いて整数対を表現するのに対して、± 3 などのより大きな値の桁集合を用いた場合への拡張表現の構成である。本稿では、この未解決となっていた JSF の拡張について考察する。さらに、通常の JSF が満たすいくつかの性質について、拡張 JSF も満足することを確認する。例えば、拡張 JSF はいくつかのルールを満たす表現として定義できることや、存在性、一意性、ハミング重みの最小性などである。

Width-3 Joint Sparse Form

Katsuyuki Okeya

Hitachi, Ltd., Systems Development Laboratory
292, Yoshida-cho, Totsuka-ku, Yokohama, 244-0817, Japan

katsuyuki.okeya.ue@hitachi.com

Abstract The joint sparse form (JSF) is a representation of a pair of integers, which is famous for accelerating a multi-scalar multiplication in elliptic curve cryptosystems. Solinas' original paper showed three unsolved problems on the enhancement of JSF. Whereas two of them have been solved, the other still remains to be done. The remaining unsolved problem is as follows: To design a represntation of a pair of integers using a larger digit set such as a set involves ± 3, while the original JSF utilizes the digit set that consists of $0, \pm 1$ for representing a pair of integers. This paper put an end to the problem. In addition, the proposed enhancement satisfies some of properties, as the original does. For example, the enhanced represetation is defined as a representation that satisfies some rules. Some other properties are the existance, the uniquness of such a representation, and the optimality of the Hamming weight.

1 楕円曲線暗号と整数の表現

楕円曲線暗号は短いビット長で高い安全性を有することで知られている。楕円曲線暗号における主要な演算としてマルチスカラー倍算がある。マルチスカラー倍算とは点 P, Q とスカラー u, v から点 $uP + vQ$ を計算する演算である。より一般的には複数個の点と同数のスカラーを用いた演算に拡張できる。一つの点と一つのスカラーを用いた場合がスカラー倍算である。

マルチスカラー倍算は楕円署名の検証で用いられる他、スカラー倍算の計算の際に櫛型

法 (comb method) や GLV 法によりマルチスカラー倍算に変換して用いられる場合もある。マルチスカラー倍算を用いる利点は、スカラー倍算を各々適用した場合と比べて、楕円二倍算の回数を削減できる点にある。

$\mathcal{D}$ をマルチスカラー倍算 $uP + vQ$ における桁集合とする。すなわち、u, v を符号付二進表現で表した際に用いる数値の組の集合である。$\mathcal{D} \subset \mathcal{D}_u \times \mathcal{D}_v$ で、$\mathcal{D}_u, \mathcal{D}_v$ はそれぞれ u, v に対する符号付二進表現に出現する数値の集合である。基本的な例としては、$\mathcal{D}_u = \mathcal{D}_v = \{0, \pm 1\}$, $\mathcal{D} = \mathcal{D}_u \times \mathcal{D}_v = \{(0,0),\ (0,\pm 1),\ (\pm 1, 0),\ (\pm 1, \pm 1),\ (\pm 1, \mp 1)\}$ である。$\mathcal{D}^{w=k} = \{(d_0, d_1) | d_0, d_1 \in \{0, \pm 1, \pm 3, \cdots, \pm 2^{k-1} \mp 1\}\}$ とする。上記の例は $\mathcal{D} = \mathcal{D}^{w=2}$ である。桁集合 $\mathcal{D}$ による符号付二進表現を $\mathcal{D}$ 表現という。

マルチスカラー倍算の計算方法の一つに Straus-Shamir 法がある。そのアルゴリズムは次の通り。なお、ECDBL は楕円二倍算、ECADD は楕円加算を表す。

Algorithm 1

Require: Points P, Q, scalars u, v.
Ensure: $uP + vQ$
1: $R_{t_1 t_2} \leftarrow t_1 P + t_2 Q, \forall (t_1, t_2) \in \mathcal{D} \setminus \{(0,0)\}$
2: $X \leftarrow \mathcal{O}$
3: **for** $j = n - 1$ down to 0
4: $\quad X \leftarrow \mathrm{ECDBL}(X)$
5: $\quad$ **if** $(u_j, v_j) \in \mathcal{D} \setminus \{(0,0)\}$
6: $\quad\quad X \leftarrow \mathrm{ECADD}(X, R_{u_j v_j})$
7: **return** X.

マルチスカラー倍算において、その計算効率は楕円加算の回数によって定まる。整数対 (u, v) $(u = (u_{n-1}, u_{n-2}, \cdots, u_0)_2,\ v = (v_{n-1},\ v_{n-2}, \cdots, v_0)_2,\ (u_j, v_j) \in \mathcal{D})$ に対して、$\mathcal{D}$ に関する共同ハミング重みとは、$JHW_{\mathcal{D}}(u, v) := \#\{j | (u_j, v_j) \neq (0,0)\}$ である。また、共同ハミング密度は $JHD_{\mathcal{D}} := JHW_{\mathcal{D}}/n$ で与えられる。

1.1 共同疎形式 (Joint Sparse Form)

桁集合が $\mathcal{D}^{w=2}$ に対する最小の共同ハミング重みを与える表現形式として、Solinas が提案した共同疎形式 (joint sparse form, JSF)[1] が知られている。

整数対 $(u^{(0)}, u^{(1)})$ に対して、JSF は次の三つの性質を満たす $\mathcal{D}^{w=2}$ 表現として定義される。

(JSF1) 連続する 3 列のうち、少なくとも一列はゼロ列

(JSF2) 非ゼロビットが連続する場合、その符号は同じ

(JSF3) $u_{j+1}^{(i)} u_j^{(i)} \neq 0$ のとき、$u_{j+1}^{(1-i)} = \pm 1$, $u_j^{(1-i)} = 0$.

この時、次の三つの性質を満たす。

(存在性) 任意の整数対に対して JSF が存在

(一意性) 任意の整数対に対して JSF は高々一つ

(最小性) JSF は最小の共同ハミング重みを持つ

Solinas の論文の最後に JSF の拡張に関するいくつかの未解決問題が示されている。

1. 左から右 (left-to-right) に生成する方法
2. 複数個 (3 つ以上) の整数に対する JSF
3. より大きな桁集合に対する JSF

(3) のより大きな桁集合への拡張については、いくつかの提案手法はあるが、いずれも最小性に関する性質が証明されていない。したがって、これは未解決問題である。

なお、$\mathcal{D}$ 表現はマルチスカラー倍算に利用されるため、最適表現を求めるアルゴリズムに対して、そこから由来する制約条件が存在する。

一つはメモリに対する要件である。マルチスカラー倍算では、$\mathcal{D}$ に対応する点を計算し、メモリに格納する。その際に要求されるメモリは、n をビット長とするとき $\#\mathcal{D} \cdot n$ である。すなわ

ちオーダーは$O(n)$である。そのため、表現変換アルゴリズムのメモリ使用量に対する要件は、オーダーが$O(n)$より小さい、もしくは$\#\mathcal{D}\cdot n$より小さいことである。

もう一つは計算量に対する要件である。マルチスカラー倍算では、1ビットごとに楕円二倍算が発生する。楕円二倍算の計算には定数回のnビット剰余乗算が必要である。したがって、全体として$O(n^3)$の計算量となる。表現変換アルゴリズムの計算量に対する要件は、オーダーが$O(n^3)$より無視できるほど小さいことである。

2 拡張共同疎形式

本節では、桁集合$\mathcal{D}^{w=3}$に対するJSFの拡張について考察する。整数対$U=(u^{(0)},u^{(1)})$に対して、次の七つの性質(EJSF1)-(EJSF7)を満たす$\mathcal{D}^{w=3}$表現を桁集合$\mathcal{D}^{w=3}$に対するJSF(EJSF)と定義する。

以下では、次の記法を用いる。

$u^{(i)}_{[j_1,j_2]}=\sum_{j=j_2}^{j_1}u^{(i)}_j2^{j-j_2}$.

$u^{(i)}_{(j_1,j_2)}=(u^{(i)}_{j_1},u^{(i)}_{j_1-1},\cdots,u^{(i)}_{j_2})$.

$U_j=\begin{bmatrix}u^{(0)}_j\\u^{(1)}_j\end{bmatrix}$, $O=\begin{bmatrix}0\\0\end{bmatrix}$.

$U_{(j_1,j_2)}=\begin{bmatrix}u^{(0)}_{j_1}, & u^{(0)}_{j_1-1}, & \cdots, & u^{(0)}_{j_2}\\u^{(1)}_{j_1}, & u^{(1)}_{j_1-1}, & \cdots, & u^{(1)}_{j_2}\end{bmatrix}$.

$\mathcal{D}^{w=3}$表現における任意の非ゼロビット$u^{(i)}_j$に対して、次の七つの条件のいずれか一つのみが成立する。

(EJSF1) $u^{(1-i)}_j=0$ かつ $U_{j+1}=U_{j+2}=O$.

(EJSF2) $u^{(1-i)}_j=0$, $U_{j+1}=O$, $u^{(0)}_{j+2}u^{(1)}_{j+2}\neq 0$.

(EJSF3) $u^{(1-i)}_j=0$, $u^{(0)}_{j+1}u^{(1)}_{j+1}\neq 0$.
$u^{(i)}_{[j+1,j]}=\pm7,\pm9$ の時は、$U_{j+3}=O$, $u^{(i)}_{j+4}=u^{(1-i)}_{j+4}\pmod 2$.

(EJSF4) $u^{(1-i)}_j\neq 0$ かつ $U_{j+1}=U_{j+2}=O$.
さらに$JHW(U_{(j+4,j+3)})\leq 1$.

(EJSF5) $u^{(1-i)}_j\neq 0$, $U_{j+1}=O$, $u^{(0)}_{j+2}u^{(1)}_{j+2}\neq 0$.
さらに$u^{(i)}_{j+2}$は(EJSF4)を満たす。(EJSF4)の形に変形不可能なこと。すなわち、$V_{(n-1,j)}\neq U_{(n-1,j)}$かつ$v^{(i')}_{[n-1,j]}=u^{(i')}_{[n-1,j]}$ $(i'=0,1)$で、$v^{(i)}_j\neq 0$に対して(EJSF4)を満たすような$V_{(n-1,j)}$が存在しないこと。

(EJSF6) $u^{(1-i)}_j\neq 0$, $U_{j+1}=U_{j+2}=O$. さらに$u^{(i)}_{[j+4,j+3]}=\pm7,\pm9$ または $u^{(1-i)}_{[j+4,j+3]}=\pm7,\pm9$. (EJSF5)の形に変形不可能なこと。

(EJSF7) $u^{(1-i)}_j\neq 0$, $U_{j+1}=O$, $u^{(0)}_{j+2}u^{(1)}_{j+2}\neq 0$.
さらに$u^{(i)}_{j+2}$は(EJSF5) (EJSF6) (EJSF7)のいずれかを満たす。(EJSF4) (EJSF5) (EJSF6)の形に変形不可能なこと。

上記の定義はうまく定義されている。すなわち、(1)二つの条件を一度に満たすようなことは起きない。(2)(EJSF7)は再帰的であるが、ビット長がnのため、(EJSF5)または(EJSF6)を満たすjが存在し、必ず完結する。また、(EJSF6)を満たす$u^{(i)}_j$は(EJSF4)に変形できないことも分かる。

さらに、この定義から、桁集合$\mathcal{D}^{w=2}$に対するJSFの定義と類似した次の条件を満たすことが分かる。

系 1 桁集合$\mathcal{D}^{w=3}$に対する*JSF*は、さらに次の条件を満たす。

(ゼロ列の分布)
連続する*3*列のうち、少なくとも一列はゼロ列である。さらに$m\geq 2$に対して、連続する$2m$列のうち、少なくともm列はゼロ列である。

(非ゼロの双対性 1)
$u^{(i)}_{j+1}u^{(i)}_j\neq 0$のとき、$u^{(1-i)}_{j+1}\neq 0$, $u^{(1-i)}_j=0$.
$u^{(1-i)}_{j+1}u^{(i)}_j\neq 0$のとき、$u^{(i)}_{j+1}\neq 0$, $u^{(1-i)}_j=0$.

(非ゼロの双対性 2)
$u^{(i)}_{j+2}u^{(i)}_j\neq 0$のとき、$u^{(1-i)}_{j+2}\neq 0$.
$u^{(1-i)}_{j+2}u^{(i)}_j\neq 0$のとき、$u^{(i)}_{j+2}\neq 0$.

桁集合$\mathcal{D}^{w=2}$に対するJSFが満たす性質は、拡張版JSFも同様に満たすことが期待される。実際、次の定理が成立つ。

定理 2 桁集合$\mathcal{D}^{w=3}$に対する*JSF(EJSF)*は、次の三つの性質を満たす。
(存在性) 任意の整数対に対して*EJSF*が存在

(一意性) 任意の整数対に対して *EJSF* は高々一つ
(最小性) *EJSF* は最小共同ハミング重みを持つ

以下の節で本定理の証明を与える。

3 存在性

本節では存在性を示す。実際、図 1 および以下に示すアルゴリズムは、任意の整数対に対して EJSF を出力する。すなわち、出力した表現は $\mathcal{D}^{w=3}$ 表現であり、(EJSF1)-(EJSF7) を満足する。

図に示すように、本アルゴリズムには A-E の五つの状態があり、初期状態は A である。各状態では、現在のビット位置 (j で表されている) から左側の列を適宜読み込み、その値に応じて変換後の列を適宜出力する。その後、状態を遷移させる。

各状態で行う変換は、次の三つの基本変換 (SW, ZF, DI) の組合せである。

- $SW(U_{(j_1,j_2)}) = \begin{bmatrix} SW(u^{(0)}_{(j_1,j_2)}) \\ SW(u^{(1)}_{(j_1,j_2)}) \end{bmatrix}$. ただし、$SW(u^{(i)}_{(j_1,j_2)}) = (0, 0, \cdots, 0, u^{(i)}_{[j_1,j_2]})$.
- $ZF(U_{(j_1,j_2)}) = \begin{bmatrix} ZF(u^{(0)}_{(j_1,j_2)}) \\ ZF(u^{(1)}_{(j_1,j_2)}) \end{bmatrix}$. ただし、$ZF(u^{(i)}_{(j_1,j_2)}) = (s, \bar{s}, \bar{s}, \cdots, \bar{s}, u^{(i)}_j - 2s, u^{(i)}_{j-1}, u^{(i)}_{j-2}, \cdots, u^{(i)}_{j_2})$, ここで $s = sgn(u^{(i)}_j)$, および $u^{(i)}_j \neq 0$ を満たす最大の j $(j_1 \geq j \geq j_2)$ である。
- $DI((u^{(i)}_{j+3}, 0, 0, \pm 3, \pm 1))$
  $= (u^{(i)}_{j+3} \pm 1, 0, 0, \mp 3, \mp 3)$.

まず、ビット列の表現を MOF に変換する。MOF とは、符号付二進表現の一つで、0 を取り除いた非ゼロビットで構成される列を考えた際に、隣り合う非ゼロビットの符号が反転する (すなわち 1 と -1 が交代で出現する) もので、最上位ビットと最下位ビットが異なるものを指す。二進表現 d に対してビット毎に $2d - d$ を計算することで得られる。

次に、変換された MOF 表現に対して、各状態において以下に示す処理を施す。初期状態においては $j = 0$ として j を初期化し、ある状態に遷移する際に j が $j \geq n$ を満たした場合に終了する。再び同じ状態に戻った際、必ず j は増加しているため、本アルゴリズムは必ず終了する。

入力は $U_{(n-1,0)}$ で与えられ、変換した表現も同じ $U_{(n-1,0)}$ に格納される。入力値を保存する場合は、入力値を出力領域に必要な分だけコピーし、変換を行えばよい。

なお、MOF 表現への変換および出力領域へのコピーは right-to-left で行うことができるため、各状態での処理に必要なビット列のみを随時変換およびコピーすることで、本アルゴリズムの処理全体を right-to-left で行うことができる。

簡便のため、$JHW(SW(U_{(j+b,j+a)}))$ を $\omega(b, a)$ と表記する。

- 状態 A: (初期状態)
  - $U_j = O$ のとき (A-1)
    1. $j \leftarrow j + 1$, $State \leftarrow A$.
  - $u^{(i)}_j \neq 0$ かつ $u^{(1-i)}_{[j+2,j]} = 0$ のとき (A-2)
    1. $U_{(j+2,j)} \leftarrow SW(U_{(j+2,j)})$.
    2. $j \leftarrow j + 3$, $State \leftarrow A$.
  - $u^{(i)}_j \neq 0$ かつ $u^{(1-i)}_{[j+2,j]} \neq 0$, $u^{(1-i)}_{[j+1,j]} = 0$ のとき (A-3)
    1. $U_{(j+2,j)} \leftarrow ZF(U_{(j+2,j)})$.
    2. $U_{(j+1,j)} \leftarrow SW(U_{(j+1,j)})$.
    3. $j \leftarrow j + 2$, $State \leftarrow B$.
  - $u^{(i)}_j \neq 0$ かつ $u^{(1-i)}_{[j+1,j]} \neq 0$, $u^{(1-i)}_j = 0$ かつ $u^{(i)}_{[j+3,j]} = \pm 7$ のとき (A-4)
    1. $State \leftarrow D$.
  - $u^{(i)}_j \neq 0$ かつ $u^{(1-i)}_{[j+1,j]} \neq 0$, $u^{(1-i)}_j = 0$ かつ $u^{(i)}_{[j+3,j]} \neq \pm 7$ のとき (A-5)
    1. $U_{(j+1,j)} \leftarrow ZF(U_{(j+1,j)})$.
    2. $j \leftarrow j + 1$, $State \leftarrow B$.
  - $u^{(i)}_j \neq 0$ かつ $u^{(1-i)}_j \neq 0$ のとき (A-6)
    1. $State \leftarrow B$.
- 状態 B: $(u^{(i)}_j u^{(1-i)}_j \neq 0)$

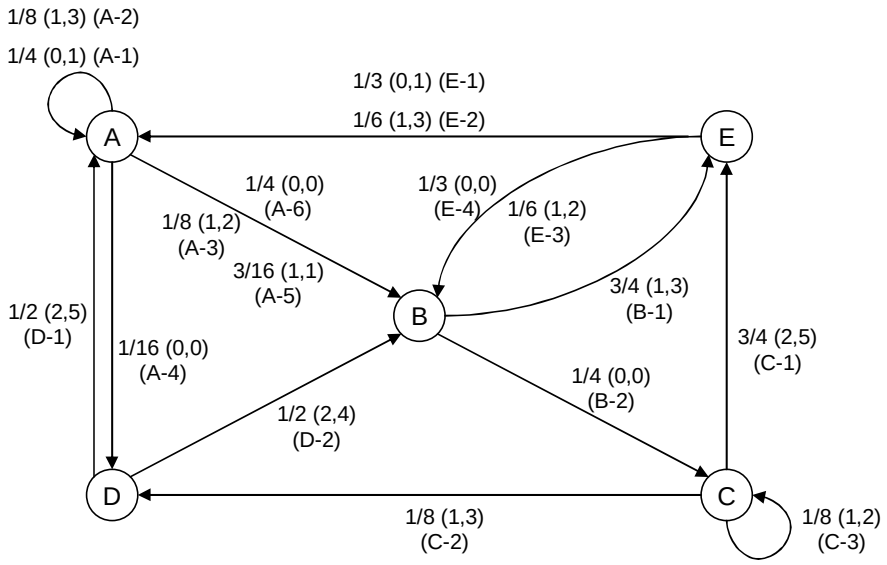

図 1: $\mathcal{D}^{w=3}$ に対する JSF の出力アルゴリズム

- $\omega(4,3) \leq 1$ のとき (B-1)
  1. $U_{(j+2,j)} \leftarrow SW(U_{(j+2,j)})$.
  2. $j \leftarrow j+3$, $State \leftarrow E$.
- $\omega(4,3) = 2$ のとき (B-2)
  1. $State \leftarrow C$.

- 状態 C: $(u_j^{(i)} u_j^{(1-i)} \neq 0, \omega(4,3) = 2)$
  - $\omega(6,5) \leq 1$ のとき (C-1)
    1. $U_{(j+2,j)} \leftarrow ZF(U_{(j+2,j)})$.
    2. $U_{(j+1,j)} \leftarrow SW(U_{(j+1,j)})$.
    3. $U_{(j+4,j+2)} \leftarrow SW(U_{(j+4,j+2)})$.
    4. $j \leftarrow j+5$, $State \leftarrow E$.
  - $\omega(6,5) = 2$, $u_{[j+6,j+3]}^{(i)} = \pm 7$ のとき (C-2)
    1. $U_{(j+2,j)} \leftarrow SW(U_{(j+2,j)})$.
    2. $j \leftarrow j+3$, $State \leftarrow D$.
  - $\omega(6,5) = 2$, $u_{[j+6,j+3]}^{(i)} \neq \pm 7$ のとき (C-3)
    1. $U_{(j+2,j)} \leftarrow ZF(U_{(j+2,j)})$.
    2. $U_{(j+1,j)} \leftarrow SW(U_{(j+1,j)})$.
    3. $j \leftarrow j+2$, $State \leftarrow C$.

- 状態 D: $(u_{[j+3,j]}^{(i)} = \pm 7)$
  - $u_{j+4}^{(1-i)} = 0$ のとき (D-1)
    1. $U_{(j+1,j)} \leftarrow ZF(U_{(j+1,j)})$.
    2. $U_{(j+3,j+1)} \leftarrow SW(U_{(j+3,j+1)})$.
    3. $u_{(j+4,j+1)}^{(i)} \leftarrow DI(u_{(j+4,j+1)}^{(i)})$ if $u_{j+4}^{(i)} \neq 0$.
    4. $j \leftarrow j+5$, $State \leftarrow A$.
  - $u_{j+4}^{(1-i)} \neq 0$ のとき (D-2)
    1. $U_{(j+1,j)} \leftarrow ZF(U_{(j+1,j)})$.
    2. $U_{(j+3,j+1)} \leftarrow SW(U_{(j+3,j+1)})$.
    3. $u_{(j+4,j+1)}^{(i)} \leftarrow DI(u_{(j+4,j+1)}^{(i)})$ if $u_{j+4}^{(i)} = 0$.
    4. $j \leftarrow j+4$, $State \leftarrow B$.

- 状態 E: $(\omega(1,0) \leq 1)$
  - $U_j = O$ のとき (E-1)
    1. $j \leftarrow j+1$, $State \leftarrow A$.
  - $u_j^{(i)} \neq 0$ かつ $u_{[j+2,j]}^{(1-i)} = 0$ のとき (E-2)
    1. $U_{(j+2,j)} \leftarrow SW(U_{(j+2,j)})$.
    2. $j \leftarrow j+3$, $State \leftarrow A$.
  - $u_j^{(i)} \neq 0$ かつ $u_{[j+2,j]}^{(1-i)} \neq 0$, $u_{[j+1,j]}^{(1-i)} = 0$ のとき (E-3)
    1. $U_{(j+2,j)} \leftarrow ZF(U_{(j+2,j)})$.
    2. $U_{(j+1,j)} \leftarrow SW(U_{(j+1,j)})$.
    3. $j \leftarrow j+2$, $State \leftarrow B$.
  - $u_j^{(i)} \neq 0$ かつ $u_j^{(1-i)} \neq 0$ のとき (E-4)
    1. $State \leftarrow B$.

4　効率性

一意性、最小性の証明の前に、本アルゴリズムの効率性 (共同ハミング密度、メモリ使用量、計算量) について確認する。

まずは、本アルゴリズムが出力する表現の共同ハミング密度について考察する。図 1 のアルゴリズムは有限マルコフ鎖として与えられている。図の矢印は状態遷移を表す。矢印の脇に示されている分数は遷移する確率であり、(x, y) は出力する列の情報で、x は出力する列の共同ハミング重み、y は出力する列の数を表す。

共同ハミング密度は有限マルコフ鎖の性質を用いて計算できる。図 1 の有限マルコフ鎖は既約 (全連結) かつ非周期的であるので、定常確率が存在する。この定常確率を線形代数を用いて計算すると次のようになる。

$$(A, B, C, D, E) = \frac{1}{423}(112, 133, 38, 47/4, 513/4)$$

次に状態遷移の際に出力される平均共同ハミング重みと平均ビット長を計算すると各々次のようになる。

$$\begin{aligned} (\text{平均共同ハミング重み}) &= \frac{563}{846} \\ (\text{平均ビット長}) &= \frac{1574}{846} \end{aligned}$$

したがって、共同ハミング密度 $JHD_{\mathcal{D}^{w=3}} = 563/1574\ (= 0.3577)$ となる。

本アルゴリズムのメモリ使用量は $O(\log n)$、計算量は $O(n)$ である。ただし、n は表現のビット長である。メモリ使用量については、入力領域と出力領域は予め与えられていると考えて、内部で使用する領域を見積もった。

5　一意性と最小性

一意性を示すには、異なる二つの表現を考え、ビットの値が異なる一番右側の位置に対して、2 べきによる剰余の値が一致することから矛盾を引出せばよい。

最小性を示すには、提案アルゴリズムが出力した U とは異なる V で、その共同ハミングウェイトが U よりも小さいものを考え、ビットの値が異なる一番右側の位置に対して、2 べきによる剰余の値が一致することから矛盾を引出す。

前提として、$u^{(i)}_{[n-1,0]} = v^{(i)}_{[n-1,0]}$ for $i = 0, 1$ が成り立つ。二つの表現は異なるので、ある j が存在し、$u^{(i)}_j \neq v^{(i)}_j$ となる。ここで、$u^{(1)}_0 \neq v^{(1)}_0$ と仮定しても一般性を失わない。$u^{(1)}_{[n-1,0]} = v^{(1)}_{[n-1,0]}$ より、$u^{(1)}_0 = v^{(1)}_0 \pmod 2$ である。$u^{(1)}_0$ が偶数であるとすると、$u^{(1)}_0 = v^{(1)}_0$ となり仮定に反する。したがって、非ゼロである。この非ゼロに着目し、この非ゼロが満たす条件 (EJSF1)-(EJSF7) ごとに矛盾を引き出す。証明は、ビット長 n に関する帰納法で行う。帰納法が可能であることは次の補題により保証される。

補題 3 (継続可能性) 提案アルゴリズムは継続可能である。すなわち、任意の状態で一時中止して初期状態から継続した場合の出力は、一時中止しないで出力したものに等しい。

また、次の補題も証明に有用である。

補題 4 (準加法性) 整数対 $(u^{(0)}, u^{(1)})$ に対する最小 *JHW(MJHW* と略記*)* は準加法性を満たす。すなわち、任意の二つの整数対 $U = (u^{(0)}, u^{(1)})$, $V = (v^{(0)}, v^{(1)})$ に対して、$MJHW(U + V) \leq MJHW(U) + MJHW(V)$ が成り立つ。さらに、$u^{(0)} = u^{(1)} \pmod 2$ の場合、任意の $d \in \{\pm 1, \pm 3\}$ に対して、$MJHW((u^{(0)} + d, u^{(1)}))$, $MJHW((u^{(0)}, u^{(1)} + d)) \geq MJHW(U)$ が成り立つ。

紙数の関係から、証明の詳細は割愛する。

参考文献

[1] Solinas, J.A., *Low-weight binary representations for pairs of integers*, Technical Report of the Centre for Applied Cryptographic Research, University of Waterloo - CACR, CORR 2001-41, 2001, available at `http://www.cacr.math.uwaterloo.ca`