Optimal Average Joint Hamming Weight and Minimal Weight Conversion of d Integers

Vorapong Suppakitpaisarn[1,2], Masato Edahiro[1,3], and Hiroshi Imai[1,2]

[1]Graduate School of Information Science and Technology, the University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo, Japan
[2]ERATO-SORST Quantum Computation and Information Project, JST
5-28-3 Hongo, Bunkyo-ku, Tokyo, Japan
[3]System IP Core Research Laboratories, NEC Corporation
1753 Shimonumabe, Nakahara-ku, Kawasaki, Japan

Abstract. In this paper, we propose the minimal joint Hamming weight conversion for any binary expansions of d integers. With redundant representations, we may represent a number by many expansions, and the minimal joint Hamming weight conversion is the algorithm to select the expansion that has the least joint Hamming weight. As the computation time of the cryptosystem strongly depends on the joint Hamming weight, the conversion can make the cryptosystem faster. Most of existing conversions are limited to some specific representations, and are difficult to apply to other representations. On the other hand, our conversion is applicable to any binary expansions. The proposed can explore the minimal average weights in a class of representation that have not been found. One of the most interesting results is that, for the expansion of integer pairs when the digit set is $\{0, \pm 1, \pm 3\}$, we show that the minimal average joint Hamming weight is 0.3575. This improves the upper bound value, 0.3616, proposed by Dahmen, Okeya, and Takagi.

Key words: Elliptic Curve Cryptography, Minimal Weight Conversion, Average Joint Hamming Weight, Digit Set Expansion

1 Introduction

The joint weight is known to affect the computation time of many operations such as the multi-scalar point multiplication of the elliptic curve cryptography,

$$K = \sum_{i=1}^{d} r_i P_i = r_1 P_1 + \cdots + r_d P_d,$$

when r_i is a natural number, and P_i is a point on the elliptic curve. With redundant representations, we can represent each r_i in more than one ways, each way, called expansion, has a different value of Hamming weight. The lower weight expansion tends to make the operation faster. Then, there are many works have explored the lower weight expansion on many specific representations [1–7].

These include the work by Solinas [1], which proposes the minimal joint weight expansion on an integer pair when the digit set is $\{0, \pm 1\}$. Also, the work by Heuberger and Muir [2, 3] presents the expansions for the digit set

$$\{-l, -(l-1), \ldots, -1, 0, 1, \ldots, u-1, u\}$$

for any natural number l, and positive integer u.

However, the minimal weight conversions of many digit sets have not been found in the literature. This causes by the fact that most of the previous works used the mathematical construction of the representation, which is hard to be found in many digit sets, for finding the conversion and the average weight.

On the other hand, we propose the conversion and the algorithm to find the average weight without concerning that mathematical construction. This enables us to find the conversions and the average weight of many interesting digit sets, in which the mathematical construction is complex. For instance, $\{0, \pm 1, \pm 3\}$ [8], that uses the same amount of memory to store the pre-computed points as $\{0, \pm 1, \pm 2\}$, but is proved to has lower minimal average weight when $d = 2$.

As our conversion is proposed for any digit sets, it might not be as fast as most of the algorithms for the specific digit set in the literatures. But, we believe that the implementer can use our algorithm as a framework, and produce more efficient algorithm for their specific digit set.

Then, we propose the algorithm to construct the Markov chain for analyzing the average joint Hamming weight from the minimal weight conversion. The Markov chain can be used for finding the minimal average weight of d integers, when

$$\{0, \pm 1, \pm \Lambda\} \subseteq Ds,$$

if Ds is the digit set, and $\Lambda = \max Ds$.

One of the most interesting is the expansion when the digit set is $\{0, \pm 1, \pm 3\}$, and $d = 2$. For this digit set, many previous works have proposed the construction, the conversion, and the analysis. They can find the upper bound for the minimal average joint Hamming weight. We can find the minimal average joint Hamming weight for this digit set, which is 0.3575. This improves the lowest upper bound in the literatures proposed by Dahmen, Okeya, and Takagi [5, 6], that is 0.3616.

To conclude, our contribution is this paper are:

1. We use the dynamic programming technique to propose the minimal weight conversion that can be applied to any digit sets, and we prove the optimality of the algorithm in Appendix B.
2. We propose the algorithm to find the minimal average joint hamming weight for a class of digit sets, and we prove the minimality of the value in Appendix C.

The remainder of this paper is organized as follows: We discuss the background knowledge of this research in Section 2. In Section 3, we propose the minimal weight conversion algorithm, with the explanation and the example.

In Section 4, we present the algorithm to construct the Markov chain used for analyzing the digit set expansion from the conversion in Section 3. Then, we use that Markov chain to find the minimal average joint Hamming weight. Last, we conclude the paper in Section 5.

2 Definition

Let Ds be the digit set mentioned in Section 1, n, d be a positive integer, $E\{Ds, d\}$ be a function from $\mathbb{Z}^d$ to $(Ds^n)^d$ such that if

$$\begin{aligned} E\{Ds, d\}(r_1, \ldots, r_d) &= \langle (e_{1,n-1}\ e_{1,n-2} \ldots e_{1,0}), \ldots, (e_{d,n-1}\ e_{d,n-2} \ldots e_{d,0}) \rangle \\ &= \langle (e_{i,t})_{t=0}^{n-1} \rangle_{i=1}^{d}, \end{aligned}$$

where

$$\sum_{t=0}^{n-1} e_{i,t} 2^t = r_i,$$

where $r_i \in \mathbb{Z}$ and $e_{i,t} \in Ds$ for all $1 \leq i \leq d$. We call $E\{Ds, d\}$ as the conversion, and we call $\langle (e_{i,t})_{t=0}^{n-1} \rangle_{i=1}^{d}$ as the expansion of $r_1, \ldots, r_d$ by the conversion $E\{Ds, d\}$. As a special case, let $E_b\{d\}$ be the binary conversion changing the integer to its binary representation where $Ds = \{0, 1\}$.

$$E_b\{1\}(12) = \langle (1100) \rangle,$$

$$E_b\{2\}(12, 21) = \langle (01100), (10101) \rangle.$$

We also define a tuple of t-th bit of r_i as,

$$E_b\{d\}(r_1, \ldots, r_d)|_t = \langle e_{1,t}, \ldots, e_{d,t} \rangle.$$

Next, we define the joint Hamming weight function of integer $r_1, \ldots r_d$ represented by the conversion $E\{Ds, d\}$, $JW_{E\{Ds,d\}}(r_1, \ldots, r_d)$, that is

$$w_t = \begin{cases} 0 & \text{if } E\{Ds, d\}(r_1, \ldots, r_d)|_t = \langle \mathbf{0} \rangle, \\ 1 & \text{otherwise} , \end{cases}$$

$$JW_{E\{Ds,d\}}(r_1, \ldots, r_d) = \sum_{t=0}^{n-1} w_t.$$

For instances, $JW_{E_b\{1\}}(12) = 2$, $JW_{E_b\{2\}}(12, 21) = 4$.

The computation time of the scalar point multiplication in Section 1 depends on the joint Hamming weight. This is because we deploy the double-and-add method, that is

$$K = \sum_{i=0}^{d} r_i P_i = 2(\ldots(2(2K_{n-1} + K_{n-2}))\ldots) + K_0,$$

where

$$K_t = \sum_{i=0}^{d} e_{i,t} P_i.$$

To complete this, we need $n-1$ point doubles, and $n-1$ point additions. However, if $E\{Ds,d\}(r_1,\ldots,r_d)|_t = \langle \mathbf{0} \rangle$, $K_t = O$. And, we need not to perform point addition on that case, as $K + O = K$. Thus, the number of point additions is $JW_{E\{Ds,d\}}(r_1,\ldots,r_d) - 1$. For instances, if

$$K = 12P_1 + 21P_2,$$

we can compute K as

$$K = 2(2(2(2P_2 + P_1) + D)) + P_2,$$

if $D = P_1 + P_2$, that has already been precomputed before the computation begins. We need 4 point doubles and 3 point additions to find the result.

If $\{0,1\} \subset Ds$, we are able to represent some number $r_i \in \mathbb{Z}$ in more than one ways. For instances, if $Ds = \{0, \pm 1\}$,

$$12 = (01100) = (10\bar{1}00) = (1\bar{1}100) = \ldots,$$

when $\bar{1} = -1$.

Let $E_m\{Ds,d\}$ be the minimal weight conversion where

$$E_m\{Ds,d\}(r_1,\ldots,r_d) = \langle e_{i,n-1} \ldots e_{i,0} \rangle_{i=1}^{t}$$

is the expansion such that for any $\langle e'_{i,n-1} \ldots e'_{i,0} \rangle_{i=1}^{t}$ where

$$\sum_{t=0}^{n-1} e_{i,t} 2^t = \sum_{t=0}^{n-1} e'_{i,t} 2^t,$$

for all $1 \leq i \leq d$,

$$\sum_{t=0}^{n-1} w'_t \geq JW_{E_m\{Ds,d\}}(r_1,\ldots,r_d),$$

and

$$w'_t = \begin{cases} 0 & \text{if } \langle e'_{1,t},\ldots,e'_{d,t} \rangle = \langle \mathbf{0} \rangle, \\ 1 & \text{otherwise}, \end{cases}$$

For instances,

$$E_m\{\{0,\pm 1\},2\}(12,21) = \langle (10\bar{1}00), (10101) \rangle,$$

$$JW_{E_m\{\{0,\pm 1\},2\}}(12,21) = 3.$$

Then, the number of point additions needed is 2. Also, we call $E_m\{Ds,d\}(r_1,\ldots,r_d)$ as the minimal weight expansion of $r_1,\ldots,r_d$ using the digit set Ds.

If $Ds_2 \subseteq Ds_1$, it is obvious that

$$JW_{E_m\{Ds_2,d\}}(r_1,\ldots,r_d) \geq JW_{E_m\{Ds_1,d\}}(r_1,\ldots,r_d).$$

Thus, we can increase the efficiency of the scalar-point multiplication by increase the size of Ds. However, the bigger Ds need more precomputation tasks. If $d = 2$, we need one precomputed point when $Ds = \{0, 1\}$, but we need 12 precomputed points when $Ds = \{0, \pm 1, \pm 3\}$.

Then, one of the contribution of this paper is to evaluate the digit set Ds. We use the average joint hamming weight defined as

$$AJW(E\{Ds,d\}) = \lim_{k\to\infty} \sum_{r_1=0}^{2^k-1} \cdots \sum_{r_d=0}^{2^k-1} \frac{JW_{E\{Ds,d\}}(r_1,\ldots,r_d)}{k2^{dk}}.$$

It is obvious that $AJW(E_b\{d\}) = 1 - \frac{1}{2^d}$. In this paper, we find the value $AJW(E_m\{Ds,d\})$ of some Ds and d. Some of these values have been found in the literatures such as

$$AJW(E_m\{\{0,\pm 1,\pm 3,\ldots,\pm(2^p-1)\},1\}) = \frac{1}{p+1}[4].$$

Also,

$$AJW(E_m\{\{-l,-(l-1),\ldots,-1,0,1,u-1,u\},d\})$$

for any positive number d,u, and natural number l, have been found by Heuberger and Muir [2, 3].

3 Minimal Weight Conversion

In this section, we propose the minimal weight conversion. It is based on the dynamic programming scheme. The input is $\langle r_1,\ldots,r_d\rangle$, and the output is $E_m\{Ds,d\}(r_1,\ldots,r_d)$, which is the minimal weight expansion of the input using the digit set Ds. The algorithm begins from the most significant bit (bit $n-1$),

$$R_{n-1} = E_b\{d\}(r_1,\ldots,r_d)|_{t=n-1} = \langle e_{1,n-1},\ldots,e_{d,n-1}\rangle,$$

and processes left-to-right to the least significant bit (bit 0),

$$R_0 = E_b\{d\}(r_1,\ldots,r_d)|_{t=0} = \langle e_{1,0},\ldots,e_{d,0}\rangle.$$

For each t ($n > t \leq 0$), we calculate minimal weight expansions of the first $n-t$ bits of the input $r_1,\ldots,r_d$ ($\left\lfloor\frac{r_1}{2^t}\right\rfloor,\ldots,\left\lfloor\frac{r_d}{2^t}\right\rfloor$) for all carry G_t defined below. We define components in our algorithm as follows:

- The *carry array* G_t is the array occured by changing the input

$$R_t = E_b\{d\}(r_1,\ldots,r_d)|_t = \langle e_{1,t},\ldots,e_{d,t}\rangle$$

to each possible output

$$R^*_t = \langle e^*_{1,t}, \ldots, e^*_{d,t} \rangle \in Ds^d.$$

Let

$$C_t = \langle c_{1,t}, \ldots, c_{d,t} \rangle$$

be the carry from bit $t-1$ to bit t, i.e. $C_0 = \langle \mathbf{0} \rangle$ and $C_n = \langle \mathbf{0} \rangle$. For each t, $1 \leq t \leq n-1$, the array C_t satifies the relation

$$R_t + C_t = R^*_t + 2C_{t+1}.$$

For instance, let the input be 1. If $-1 \in Ds$, we can output -1 in this bit, and carry 1 to more significant bit, because

$$1 = 1 \times 2 + (-1).$$

If $3 \in Ds$, we can carry -1 to more significant bit and output 3, as

$$1 = -1 \times 2 + 3.$$

We define the carry set Cs as the set of possible carry. When the digit set $Ds = \{0, \pm 1, \pm 3\}$, the carry set is $Cs = \{0, \pm 1, \pm 2, \pm 3\}$. We propose the algorithm to find the set Cs in Appendix A.

– The *minimal weight array* w_t is the array of the positive integer w_{t,G_t} for any $G_t \in Cs^d$. The integer w_{t,G_t} is the minimal joint weight of the first $n-t$ bits of the input $r_1, \ldots, r_d$ ($\left\lfloor \frac{r_1}{2^t} \right\rfloor, \ldots, \left\lfloor \frac{r_d}{2^t} \right\rfloor$) when we carry $G_t = \langle g_{i,t} \rangle_{i=1}^d$, e.g.

$$w_{t,G_t} = JW_{E_m\{Ds,d\}}(\left\lfloor \frac{r_1}{2^t} \right\rfloor + g_{1,t}, \ldots, \left\lfloor \frac{r_d}{2^t} \right\rfloor + g_{d,t}).$$

– The *subsolution array* Q_t is the array of the string $Q_{t,\langle i,G_t \rangle}$ for any $1 \leq i \leq d$ and $G_t \in Cs^d$. Each $Q_{t,\langle i,G_t \rangle}$ represents the minimal weight expansion of the first $n-t$ bits of the input $r_1, \ldots, r_d$ when we carry $G_t = \langle g_{i,t} \rangle_{i=1}^d$, e.g.

$$w_{t,G_t} = E_m\{Ds, d\}(\left\lfloor \frac{r_1}{2^t} \right\rfloor + g_{1,t}, \ldots, \left\lfloor \frac{r_d}{2^t} \right\rfloor + g_{d,t}).$$

We note that the length of the string $Q_{t,\langle i,G_t \rangle}$ is $n-t$, and w_{t,G_t} is the joint Hamming weight of the string $Q_{t,\langle 1,G_t \rangle}$, ..., $Q_{t,\langle d,G_t \rangle}$. There may exist some $g_{i,t} \in Ds$ such that $\left\lfloor \frac{r_1}{2^t} \right\rfloor + g_{i,t}$ can not be represented using the string length $n-t$ of Ds. In that case, we represent $Q_{t,\langle i,G_t \rangle}$ with the null string, and assign w_{t,G_t} to ∞.

When we process the bit t, we find the minimal weight array w_t and the subsolution array Q_t from the input

$$R_t = E_b\{d\}(r_1, \ldots, r_d)|_t = \langle e_{1,t}, \ldots, e_{d,t} \rangle$$

and the minimal weight array w_{t+1} , and the subsolution array Q_{t+1}.

We define the function

$$MW : \mathbb{Z}^{|Cs^d|} \times ((Ds^{n-t})^d)^{|Cs^d|} \times \{0,1\}^d \times Cs^d \to \mathbb{Z} \times ((Ds^{n-t+1})^d)$$

such that

$$(w_{t,G_t}, Q_{t,G_t}) = MW(w_{t+1}, Q_{t+1}, R_t, G_t),$$

where w_{t+1} is the minimal weight array from bit $n-1$ to bit $t+1$, and w_t is the minimal weight array from bit $n-1$ to bit t. Q_{t+1} is the subsolution string from bit $n-1$ to bit $t+1$, and Q_t is the subsolution string from bit $n-1$ to bit t. Since $w_t = \langle w_{t,G_t}\rangle_{G_t \in Cs^d}$ and $Q_t = \langle Q_{t,G_t}\rangle_{G_t \in Cs^d}$, we also define

$$(w_t, Q_t) = MW(w_{t+1}, Q_{t+1}, R_t).$$

It is important to note that w_t is only depend on w_{t+1} and R_t, we use only 2 arrays w and lw to represent w_t and w_{t+1} to reduce memory consumption. Similarly, we store Q_t and Q_{t+1} by Q and lQ.

When we implement this idea, we use 2 arrays to represent w_t and w_{t+1}, called w and lw respectively. We store the array w_{t+1} in lw and use the array to compute the array w, which is used for representing w_t. After that, we replace the array lw with w, decrease t by 1, and continue calculating the subsolution for less significant bits. Similarly, we store Q_t and Q_{t+1} by Q and lQ.

Example 1. Compute the minimal weight expansion of 3 and 7 when the digit set is $\{0, \pm1, \pm3\}$, $E_m\{\{0, \pm1, \pm3\}, 2\}(3, 7)$. Note that $E_b\{2\}(3, 7) = \langle (011), (111)\rangle$.

– *Step 1* Consider the most significant bit, the input

$$R_2 = E_b\{2\}(3, 7)|_{t=2} = \langle 0, 1\rangle.$$

There is the carry from less significant bit. Table 1 shows the Hamming weight of the most significant bit for each carry $G_2 \in Cs^2$, lw_{G_2}. To summarize the result from the table, if $G_2 = \langle 0, -1\rangle$, the input with the carry from less significant bit,

$$R_2 + G_2 = \langle 0, 1\rangle + \langle 0, -1\rangle = \langle 0, 0\rangle,$$

and the Hamming weight $w_{2,\langle 0,-1\rangle} = 0$.
If $G_2 = \langle 1, 0\rangle$, the input with the carry,

$$R_2 + G_2 = \langle 0, 1\rangle + \langle 1, 0\rangle = \langle 1, 1\rangle,$$

and $w_{2,\langle 1,0\rangle} = 1$. The Hamming weight $w_{2,G}$ is 1 for any G, such that

$$R_2 + G_2 \in Ds^d - \{\langle \mathbf{0}\rangle\}.$$

If $G_2 = \langle 0, 1\rangle$,

$$R_2 + G_2 = \langle 0, 1\rangle + \langle 0, 1\rangle = \langle 0, 2\rangle,$$

and $w_{2,\langle 0,1\rangle} = \infty$. The Hamming weight $w_{2,G}$ is ∞ for any G_2, such that $R_2 + G_2 \notin Ds^d$.
As we want to keep the length of the bit string unchanged, we do not generate the carry from the most significant bit. Then, we need to represent the input with the carry, R_2+G_2 using the single bit. As a result, the hamming weight, $w_{2,G}$ is 0 if $R_2 + G_2 = \langle \mathbf{0} \rangle$. $w_{2,G}$ is ∞ if $R_2 + G_2 \notin Ds^d$. And, $w_{2,G}$ is 1 otherwise.

– *Step 2* Next, we consider bit 1. In this bit,

$$R_1 = E_b\{2\}(3,7)|_{t=1} = \langle 1,1\rangle.$$

Consider the case when the carry from the least significant bit $G_1 = \langle 1,0\rangle$. Then, $R_1+G_1 = \langle 2,1\rangle$. There are 4 ways to write $\langle 2,1\rangle$ in the form $2G_{t+1}+R_t^*$ where $G_{t+1} \in Cs^d$ is the carry to the most significant bit and $R_t^* \in Ds^d$ is the candidate for the output. That is

$$\begin{aligned}\langle 2,1\rangle &= 2\times\ \langle 1,0\rangle\ +\ \langle 0,1\rangle \\ &= 2\times \langle 1,-1\rangle +\ \langle 0,3\rangle \\ &= 2\times\ \langle 1,1\rangle\ + \langle 0,-1\rangle \\ &= 2\times\ \langle 1,2\rangle\ + \langle 0,-3\rangle\end{aligned}$$

If we output R_t^* for this bit, the Hamming weight is

$$w_{t,G_t} = \min_{G_{t+1},R_t^*}[w_{t+1,G_{t+1}} + JW(R_t^*)],$$

when $JW(D) = 0$ if $D = \langle \mathbf{0}\rangle$, $JW(D) = 1$ otherwise. From Table 1,

$$w_{2,\langle 1,0\rangle} = w_{2,\langle 1,-1\rangle} = w_{2,\langle 1,2\rangle} = 1,$$

$$w_{2,\langle 1,1\rangle} = \infty.$$

And,

$$JW(\langle 1,0\rangle) = JW(\langle 0,3\rangle) = JW(\langle 0,-1\rangle) = JW(\langle 0,-3\rangle) = 1.$$

Then,

$$w_{1,\langle 1,0\rangle} = \min_{G_2,R_t^*}[w_{2,G_2} + JW(R_1^*)] = 1+1 = 2.$$

We show the array w_{1,G_1} on this bit in Table 2.

– *Step 3* On the least significant bit, the input $R_0 = \langle 1,1\rangle$. The value $w_{0,\langle 0,0\rangle}$ is the minimal Hamming weight, and we need not to compute w_{0,G_0} for $G_0 \neq \langle \mathbf{0}\rangle$. When $G_0 = \langle 0,0\rangle$, $R_0 + G_0 = \langle 1,1\rangle$. Similar to bit 1, we find

$$w_{0,\langle 0,0\rangle} = \min_{G_1,R_0^*}[w_{1,G_1} + JW(R_0^*)],$$

such that $2\times G_1 + R_0^* = \langle 1,1\rangle$, and $G_1 \in Cs^d$, $R_0^* \in Ds^d$. We show the value of each possible G_1, R_0^* with w_{1,G_1}, $JW(R_0^*)$, and $w_{1,G_1} + JW(R_0^*)$ in Table 3. Shown in the table, the minimal Hamming weight is

$$\min_{G_1,R_0^*}[w_{1,G_1} + JW(R_0^*)] = 2.$$

Table 1. The minimal Hamming weight of the most significant bit, $w = w_{2,G_2}$, when the input bit $R_2 = \langle 0, 1\rangle$

G_2	w	G_2	w	G_2	w	G_2	w	G_2	w	G_2	w	G_2	w
$\langle -3,-3\rangle$	∞	$\langle -2,-3\rangle$	∞	$\langle -1,-3\rangle$	∞	$\langle 0,-3\rangle$	∞	$\langle 1,-3\rangle$	∞	$\langle 2,-3\rangle$	∞	$\langle 3,-3\rangle$	∞
$\langle -3,-2\rangle$	1	$\langle -2,-2\rangle$	∞	$\langle -1,-2\rangle$	1	$\langle 0,-2\rangle$	1	$\langle 1,-2\rangle$	1	$\langle 2,-2\rangle$	∞	$\langle 3,-2\rangle$	1
$\langle -3,-1\rangle$	1	$\langle -2,-1\rangle$	∞	$\langle -1,-1\rangle$	1	$\langle 0,-1\rangle$	0	$\langle 1,-1\rangle$	1	$\langle 2,-1\rangle$	∞	$\langle 3,-1\rangle$	1
$\langle -3,0\rangle$	1	$\langle -2,0\rangle$	∞	$\langle -1,0\rangle$	1	$\langle 0,0\rangle$	1	$\langle 1,0\rangle$	1	$\langle 2,0\rangle$	∞	$\langle 3,0\rangle$	1
$\langle -3,1\rangle$	∞	$\langle -2,1\rangle$	∞	$\langle -1,1\rangle$	∞	$\langle 0,1\rangle$	∞	$\langle 1,1\rangle$	∞	$\langle 2,1\rangle$	∞	$\langle 3,1\rangle$	∞
$\langle -3,2\rangle$	1	$\langle -2,2\rangle$	∞	$\langle -1,2\rangle$	1	$\langle 0,2\rangle$	1	$\langle 1,2\rangle$	1	$\langle 2,2\rangle$	∞	$\langle 3,2\rangle$	1
$\langle -3,3\rangle$	∞	$\langle -2,3\rangle$	∞	$\langle -1,3\rangle$	∞	$\langle 0,3\rangle$	∞	$\langle 1,3\rangle$	∞	$\langle 2,3\rangle$	∞	$\langle 3,3\rangle$	∞

Table 2. The minimal Hamming weight of bit 1, $w = w_{1,G_1}$, when the input bit $R_1 = \langle 1, 1\rangle$, and the array w_{1,G_1} of the most significant bit is shown in Table 1

G_1	w	G_1	w	G_1	w	G_1	w	G_1	w	G_1	w	G_1	w
$\langle -3,-3\rangle$	1	$\langle -2,-3\rangle$	1	$\langle -1,-3\rangle$	0	$\langle 0,-3\rangle$	1	$\langle 1,-3\rangle$	1	$\langle 2,-3\rangle$	1	$\langle 3,-3\rangle$	∞
$\langle -3,-2\rangle$	2	$\langle -2,-2\rangle$	1	$\langle -1,-2\rangle$	1	$\langle 0,-2\rangle$	1	$\langle 1,-2\rangle$	2	$\langle 2,-2\rangle$	1	$\langle 3,-2\rangle$	∞
$\langle -3,-1\rangle$	1	$\langle -2,-1\rangle$	2	$\langle -1,-1\rangle$	1	$\langle 0,-1\rangle$	2	$\langle 1,-1\rangle$	1	$\langle 2,-1\rangle$	2	$\langle 3,-1\rangle$	∞
$\langle -3,0\rangle$	2	$\langle -2,0\rangle$	1	$\langle -1,0\rangle$	1	$\langle 0,0\rangle$	1	$\langle 1,0\rangle$	2	$\langle 2,0\rangle$	1	$\langle 3,0\rangle$	∞
$\langle -3,1\rangle$	∞	$\langle -2,1\rangle$	∞	$\langle -1,1\rangle$	∞	$\langle 0,1\rangle$	∞	$\langle 1,1\rangle$	∞	$\langle 2,1\rangle$	∞	$\langle 3,1\rangle$	∞
$\langle -3,2\rangle$	2	$\langle -2,2\rangle$	2	$\langle -1,2\rangle$	2	$\langle 0,2\rangle$	2	$\langle 1,2\rangle$	2	$\langle 2,2\rangle$	2	$\langle 3,2\rangle$	∞
$\langle -3,3\rangle$	1	$\langle -2,3\rangle$	2	$\langle -1,3\rangle$	1	$\langle 0,3\rangle$	2	$\langle 1,3\rangle$	1	$\langle 2,3\rangle$	2	$\langle 3,3\rangle$	∞

Table 3. List of possible G_1, R_0^* such that $2 \times G_1 + R_0^* = \langle 1, 1\rangle$ and $G_1 \in \{0, \pm 1, \pm 2, \pm 3\}^2$, $R_0^* \in \{0, \pm 1, \pm 3\}^2$, with w_{1,G_1} (refer to Table 2), $JW(R_0^*)$, $w_{1,G_1} + JW(R_0^*)$ of each G_1,R_0^*

G_1	R_0^*	w_{1,G_1}	$JW(R_0^*)$	$w_{1,G_1} + JW(R_0^*)$
$\langle -1,-1\rangle$	$\langle 3,3\rangle$	1	1	2
$\langle -1,0\rangle$	$\langle 3,1\rangle$	1	1	2
$\langle -1,1\rangle$	$\langle 3,-1\rangle$	∞	1	∞
$\langle -1,2\rangle$	$\langle 3,-3\rangle$	2	1	3
$\langle 0,-1\rangle$	$\langle 1,3\rangle$	2	1	3
$\langle 0,0\rangle$	$\langle 1,1\rangle$	1	1	2
$\langle 0,1\rangle$	$\langle 1,-1\rangle$	∞	1	∞
$\langle 0,2\rangle$	$\langle 1,-3\rangle$	2	1	3
$\langle 1,-1\rangle$	$\langle -1,3\rangle$	1	1	2
$\langle 1,0\rangle$	$\langle -1,1\rangle$	2	1	3
$\langle 1,1\rangle$	$\langle -1,-1\rangle$	∞	1	∞
$\langle 1,2\rangle$	$\langle -1,-3\rangle$	2	1	3
$\langle 2,-1\rangle$	$\langle -3,3\rangle$	2	1	3
$\langle 2,0\rangle$	$\langle -3,1\rangle$	1	1	2
$\langle 2,1\rangle$	$\langle -3,-1\rangle$	∞	1	∞
$\langle 2,2\rangle$	$\langle -3,-3\rangle$	2	1	3

We show the algorithm in detail Algorithm 1,2. The algorithm can be described as follows:

- We set the array $w_{n-1} = \langle w_{n-1,G_{n-1}} \rangle_{G_{n-1} \in Cs^d}$ in Algorithm 1 Line 2. We set $w_{n-1,\langle \mathbf{0} \rangle} \leftarrow 0$ and $w_{n-1,G_{n-1}} \leftarrow \infty$ for any $G \neq \langle \mathbf{0} \rangle$. The reason behind this is that we do not carry any things from the most significant bit to keep the length of the bit string unchanged.
- We define the array $Q_t = \langle Q_{t+1,\langle i,G \rangle} \rangle$ for $1 \leq i \leq d$, and $G \in Cs^d$ in Algorithm 1 Line 3. Obviously, all $Q_{n-1,\langle i,G \rangle}$ are set to the null string.
- The size of the array w_t and Q_t is equal to $||Cs||^d$ and $dn||Cs||^d$ respectively. That number makes the memory required by our algorithms larger than the previous works. As this algorithm is generalized for any digit sets, further optimization is difficult. We suggest implementers to make the array size lower when they implement the method on their specific digit set.
- Shown in Algorithm 1 Lines 4-7, we run the algorithm from left to right (the most significant bit to the least significant bit). Left-to-right algorithms is said to be faster than right-to-left algorithms, as the more significant bits usually arrive to the system before. However, Algorithm 1,2 is not online; as it cannot produce the subsolution before all input bits arrive.
- The loop in Algorithm 2 explores all the possible values of the carry tuple $G_t \in Cs^d$ from less significant bits. As a result, we get $we = \langle we_{R_t^*} \rangle_{R_t^* \in Ds^d}$. we_E shows the minimal Hamming weight in the case that the carry tuple is $G_t = \langle g_{i,t} \rangle_{i=1}^d$, and the output is $R_t^* = \langle r_{i,t}^* \rangle_{i=1}^d$. Each $we_{R_t^*}$ is assigned at the loop in Algorithm 2 Lines 3-10. In Line 5, we compute the array G_{t+1} as the carry to more significant bits, in the case that we output R_t^*. And in Line 6, $we_{R_t^*}$ is assigned to $w_{t+1,G_{t+1}}$ if $R_t^* = \langle \mathbf{0} \rangle$, and $w_{t+1,G_{t+1}} + 1$ otherwise.
- In Algorithm 2 Line 11, we select the output for each carry tuple G_t, which produce the minimal weight substring. The output is $EA = \langle ea_i \rangle_{i=1}^d \in Ds^d$ such that we_{EA} has the minimal value among we. We define the array CE as the carry to more significant bits, in the case that we output EA in Line 15. Obviously, the minimal weight of the suboutput w_{t,G_t} is equal to we_{EA}, and the suboutput $Q_{t,\langle i,G_t \rangle}$ is the concatenation of the subsolution of more significant bits when the carry is CE, $Q_{t+1,\langle i,CE \rangle}$, with EA.
- We define the solution in Algorithm 1 Line 8. Obviously, the solution is the one from the least significant bit when the carry tuple is $\langle \mathbf{0} \rangle$.

Example 2. Compute $E_m\{\{0, \pm 1, \pm 3\}, 2\}(23, 5)$ using Algorithm 1,2.

- $E_b\{2\}(23, 5) = \langle (10111), (00101) \rangle$.
- When $Ds = \{0, \pm 1, \pm 3\}$, $Cs = \{0, \pm 1, \pm 2, \pm 3\}$.
- To simplify the explanation, we present it when the loop in Algorithm 1 Line 4-6 assigned t to 0, that is the last time on this loop. This means we have computed w_1 and Q_1. In this example, $w_t = w_{t,G_t}$ where $G_t \in \{0, \pm 1, \pm 2, \pm 3\}^2$. As w_1, Q_1 has 49 elements, we are not able to list them all. To show some elements of w_1, Q_1,

$$w_{1,\langle 0,0 \rangle} = 3, w_{1,\langle 1,0 \rangle} = 2, w_{1,\langle 2,0 \rangle} = 3.$$

Algorithm 1 Minimum joint weight conversion to any digit sets Ds in the binary expansion

```
Require: r_1, ..., r_d
    The desired digit set Ds
Ensure: E_m{Ds, d}(r_1, ..., r_d)
  Let Cs be a carry set such that for all c ∈ Cs and d ∈ Ds, (c+d)/2, (c+d+1)/2 ∈ Cs.
  Let w_t be an array of w_{t,G_t} for any G_t ∈ Cs^d.
    w_{n,G_n} ← 0 if G_{n-1} = ⟨0⟩.
    w_{n,G_n} ← ∞ otherwise.
  Let Q_t ← ⟨Q_{t,⟨i,G_t⟩}⟩ for any 1 ≤ i ≤ d and G_t ∈ Cs^d.
    All Q_{n,⟨i,G_t⟩} are initiated to a null string.
  for t ← n − 1 to 0 do
    R_t ← E_b{d}(r_1, ..., r_d)|_t.
    (w_t, Q_t) ← MW(w_{t+1}, Q_{t+1}, R_t) (We define the function MW in Algorithm 2)
  end for
  Let Z ← ⟨0⟩.
    E_m{Ds, d}(r_1, ..., r_d) ← ⟨Q_{0,⟨i,Z⟩}⟩_{i=1}^d
```

$$Q_{1,\langle 1,\langle 0,0\rangle\rangle} = (1011), Q_{1,\langle 1,\langle 1,0\rangle\rangle} = (0300), Q_{1,\langle 1,\langle 2,0\rangle\rangle} = (0301).$$

$$Q_{1,\langle 2,\langle 0,0\rangle\rangle} = (0010), Q_{1,\langle 2,\langle 1,0\rangle\rangle} = (0010), Q_{1,\langle 2,\langle 2,0\rangle\rangle} = (0010).$$

- Although, the loop in Algorithm 2 examines all $G_0 \in Cs^2$, we focus our interested the step where $G_0 = \langle \mathbf{0} \rangle$. Note that in this case
$$AE \leftarrow \langle 1,1\rangle + \langle 0,0\rangle = \langle 1,1\rangle.$$
- Now, we focus our interested to the loop in Algorithm 2 Line 3-10. If $R_0^* = \langle 0,0\rangle$, $ae_1 - r_{0,1}^* = 1$ and $2 \nmid (ae_1 - r_{0,1}^*)$. Then, $we_{\langle 0,0\rangle} \leftarrow \infty$.
- If $R_0^* = \langle 1,1\rangle$,
$$G_1 \leftarrow \langle \frac{ae_1 - r_{0,1}^*}{2}, \frac{ae_2 - r_{0,2}^*}{2} \rangle = \langle 0,0\rangle.$$
As stated on the first paragraph, $w_{1,\langle 0,0\rangle} = 3$. Then, $we_{\langle 1,1\rangle} \leftarrow 3 + 0 = 3$ by Line 6.
- If $R_0^* = \langle -1,-3\rangle$,
$$G_1 \leftarrow \langle \frac{ae_1 - r_{0,2}^*}{2}, \frac{ae_1 - r_{0,2}^*}{2} \rangle = \langle 1,2\rangle.$$
Then, we refer to $w_{1,\langle 1,2\rangle}$ which is 1. Then, $we_{\langle -1,-3\rangle} \leftarrow 1 + 1 = 2$.
- In Line 11, we select the least number among we, and the minimum value is $we_{\langle -1,-3\rangle} = 2$. Then, $w_{0,\langle 0,0\rangle} = 2$.
$$Q_{0,\langle 1,\langle 0,0\rangle\rangle} \leftarrow \langle Q_{1,\langle 1,\langle 1,2\rangle\rangle}, -1\rangle = (0300\bar{1}).$$
$$Q_{0,\langle 2,\langle 0,0\rangle\rangle} \leftarrow \langle Q_{1,\langle 2,\langle 1,2\rangle\rangle}, -3\rangle = (0100\bar{3}),$$
which is the output of the algorithm.

Algorithm 2 Function MW compute the subsolution for bit t given the subsolution of bit $t+1$ and the input in bit t

Require: The minimal weight array of more significant bits w_{t+1}, the subsolution of more significant bits Q_{t+1}, and the input R_t
Ensure: The minimal weight array w_t and the subsolution Q_t
for all $G_t = \langle g_{i,t} \rangle_{i=1}^{d} \in Cs^d$ **do**
  $AE = \langle ae_i \rangle_{i=1}^{d} \leftarrow R_t + G_t$
  for all $R_t^* = \langle r_{i,t}^* \rangle_{i=1}^{d} \in Ds^d$ **do**
    if $2|(ae_i - r_{i,t}^*)$ for all $1 \leq i \leq d$ **then**
      $G_{t+1} \leftarrow \langle \frac{ae_i - r_{i,t}^*}{2} \rangle_{i=1}^{d}$
      $we_{R_t^*} \leftarrow w_{t+1,G_{t+1}}$ if $G_{t+1} = \langle \mathbf{0} \rangle$.
      $we_{R_t^*} \leftarrow w_{t+1,G_{t+1}} + 1$ otherwise.
    else
      $we_{R_t^*} \leftarrow \infty$
    end if
  end for
  Let we_{EA} is the minimal value among we.
  $w_{t,G_t} \leftarrow we_{EA}$
  Let $EA = \langle ea_i \rangle_{i=1}^{d}$.
  $CE = \langle ce_i \rangle_{i=1}^{d} \leftarrow \langle \frac{ae_i - ea_i}{2} \rangle_{i=1}^{d}$
  $Q_{t,\langle i,G_t \rangle} \leftarrow \langle Q_{t+1,\langle i,CE \rangle}, ea_i \rangle$ for all $1 \leq i \leq d$
end for

4 Analysis

In this section, we propose the algorithm to analyze the average joint Hamming weight for each digit set. For this purpose, we propose a Markov chain where its states are minimal weight arrays w, and transition is function MW. As we will focus on only the joint Hamming weight without regarding which bit we are computing, we represent w_{t+1}, w_t with lw, w respectively. Also, we refer G_t as G.

As we have seen in the previous section, we do not have to consider Q in function MW when we are interested only the Hamming weight. Then, we can redefine the function MW as

$$MW : \mathbb{Z}^{|Cs^d|} \times \{0,1\}^d \rightarrow \mathbb{Z}^{|Cs^d|},$$

$$w_y = MW(w_x, R).$$

4.1 Analysis Method

From Algorithm 1,2, we propose Algorithm 3 to construct the Markov chain

$$A = (Q_A, \Sigma, \sigma_A, I_A, P_A),$$

where

- Q_A is a set of states,
- Σ is the alphabet,
- $\sigma_A \subseteq Q_A \times \Sigma \times Q_A$ is a set of transitions,
- $I_A : Q_A \rightarrow \mathbb{R}^+$ is an initial-state probabilities for each state in Q_A,
- $P_A : \sigma_A \rightarrow \mathbb{R}^+$ is the transition probabilities for each transition in σ_A.

The algorithm is described as follows:

- We define the set Q_A as the set of equivalence classes of the possible value of w_t in Algorithm 1,2. Let $w_x = \langle w_{x,G} \rangle_{G \in Cs^d}$ and $w_{x'} = \langle w_{x',G} \rangle_{G \in Cs^d}$ be the possible value of w_t. We consider w_x and $w_{x'}$ equivalent if and only if
$$\exists p \forall G (w_{x,G} + p = w_{x',G})$$
when $p \in \mathbb{Z}$ and $G \in Cs^d$. In Appendix C, we show the proof that the number of equivalence classes is finite when
$$\{0, \pm 1, \pm \Lambda\} \subseteq Ds,$$
if $\Lambda = \max Ds$.
- The number of states becomes very large when the digit set becomes larger. For example, the number of states is $1,216,376$ for $d = 3$ and
$$Ds = \{0, \pm 1, \pm 3\}.$$
This makes us unable to find the average joint Hamming weight in many cases.
- To find the average joint Hamming weight, we need to find the possibility that the Markov chain is on each equivalence class after we input a bit string length $n \rightarrow \infty$. That is the stationary distribution of the Markov chain. We consider the function MW, defined in Algorithm 2, as the transition from the equivalence class of w_x to the equivalence class of w_y, where the input of the transition is R. It is obvious that if w_x is equivalent w'_x and
$$w_y = MW(w_x, R),$$
$$w_{y'} = MW(w'_x, R),$$
w_α and w_β are equivalent. Then, the transition is well-defined. By this definition,
$$\Sigma = \{0, 1\}^d,$$
as in Line 1 of Algorithm 3. Also, the set of transition σ_A is defined as
$$\sigma_A = \{(w_x, R, w_y) \in Q_A \times \Sigma \times Q_A | w_y = MW(w_x, R)\}.$$
- We initiate w_t in Algorithm 1 Line 2. We refer the value initiated to w_t as w_I, as shown in Line 3 of Algorithm 3. We set the value w_I as the initial state of the Markov chain. By the definition of I_A, $I_A(w_I) = 1$, and $I_A(w) = 0$ if $w \neq w_I$, as shown in Algorithm 3 Line 18.

– We generate the set of state Q_A using the algorithm based on the breadth-first search scheme starting from w_I. This is shown in Algorithm 3 Lines 5-17.
– Since the occurence possibility of all alphabets is equal, the transform probability $P_A(\gamma) = \frac{1}{|\Sigma|}$ for all $\gamma \in \sigma_A$. This is shown in Algorithm 3 Line 11.

We illustrate how Algorithm 3 works with Example 3,4,5.

Algorithm 3 Construct the Markov chain used for finding the average minimal weight

Require: the digit set Ds
The number of scalars d
Ensure: Markov chain $A = (Q_A, \Sigma, \sigma_A, I_A, P_A)$
1: $\Sigma \leftarrow \{0,1\}^d$, $Q_A \leftarrow \oslash$, $\sigma_A \leftarrow \oslash$
2: Cs : carry set for Ds
3: $w_I \leftarrow \langle w_{I,G} \rangle_{G \in Cs^d}$, where
$w_{I,\langle \mathbf{0} \rangle} \leftarrow 0$ and $w_{I,G} \leftarrow \infty$ otherwise
4: $Qu \leftarrow \{w_I\}$
5: **while** $Qu \neq \oslash$ **do**
6: let $\pi \in Qu$
7: $w_x \leftarrow \pi$, $Qu \leftarrow Qu - \pi$
8: **for all** $R \in \Sigma$ **do**
9: $w_y \leftarrow MW(w_x, R)$
10: $\sigma_A \leftarrow \sigma_A \cup \{(w_x, R, w_y)\}$
11: $P_A(w_x, R, w_y) \leftarrow \frac{1}{|\Sigma|}$
12: **if** $w_y \notin Q_A$ and $w_y \neq w_x$ **then**
13: $Qu \leftarrow Qu \cup \{w_y\}$
14: **end if**
15: **end for**
16: $Q_A \leftarrow Q_A \cup \{w_x\}$
17: **end while**
18: $I_A(w) \leftarrow 1$ if $w = w_I$, $I_A(w) \leftarrow 0$ otherwise.

Example 3. Let $d = 2$ and $Ds = \{0, \pm 1, \pm 3\}$, $Cs = \{0, \pm 1, \pm 2, \pm 3\}$ and

$$\begin{aligned} w = {} & \langle w \rangle_{G \in Cs^2} \\ = {} & \langle w_{\langle -3,-3 \rangle}, w_{\langle -3,-2 \rangle}, w_{\langle -3,-1 \rangle}, w_{\langle -3,0 \rangle}, w_{\langle -3,1 \rangle}, w_{\langle -3,2 \rangle}, w_{\langle -3,3 \rangle}, \\ & w_{\langle -2,-3 \rangle}, w_{\langle -2,-2 \rangle}, w_{\langle -2,-1 \rangle}, w_{\langle -2,0 \rangle}, w_{\langle -2,1 \rangle}, w_{\langle -2,2 \rangle}, w_{\langle -2,3 \rangle}, \\ & w_{\langle -1,-3 \rangle}, w_{\langle -1,-2 \rangle}, w_{\langle -1,-1 \rangle}, w_{\langle -1,0 \rangle}, w_{\langle -1,1 \rangle}, w_{\langle -1,2 \rangle}, w_{\langle -1,3 \rangle}, \\ & w_{\langle 0,-3 \rangle}, w_{\langle 0,-2 \rangle}, w_{\langle 0,-1 \rangle}, w_{\langle 0,0 \rangle}, w_{\langle 0,1 \rangle}, w_{\langle 0,2 \rangle}, w_{\langle 0,3 \rangle}, \\ & w_{\langle 1,-3 \rangle}, w_{\langle 1,-2 \rangle}, w_{\langle 1,-1 \rangle}, w_{\langle 1,0 \rangle}, w_{\langle 1,1 \rangle}, w_{\langle 1,2 \rangle}, w_{\langle 1,3 \rangle}, \\ & w_{\langle 2,-3 \rangle}, w_{\langle 2,-2 \rangle}, w_{\langle 2,-1 \rangle}, w_{\langle 2,0 \rangle}, w_{\langle 2,1 \rangle}, w_{\langle 2,2 \rangle}, w_{\langle 2,3 \rangle}, \\ & w_{\langle 3,-3 \rangle}, w_{\langle 3,-2 \rangle}, w_{\langle 3,-1 \rangle}, w_{\langle 3,0 \rangle}, w_{\langle 3,1 \rangle}, w_{\langle 3,2 \rangle}, w_{\langle 3,3 \rangle} \rangle. \end{aligned}$$

Then, the initial state $w = w_I \in Q_A$ is

$$\begin{array}{l}\langle\infty,\infty,\infty,\infty,\infty,\infty,\ \infty,\\ \infty,\ \infty,\infty,\infty,\infty,\infty,\ \infty,\\ \infty,\ \infty,\infty,\infty,\infty,\infty,\ \infty,\\ \infty,\ \infty,\infty,\ 0,\ \infty,\infty,\ \infty,\\ \infty,\ \infty,\infty,\infty,\infty,\infty,\ \infty,\\ \infty,\ \infty,\infty,\infty,\infty,\infty,\ \infty,\\ \infty,\ \infty,\infty,\infty,\infty,\infty,\infty\rangle.\end{array}$$

Refered to Example 1, if $w_{t+1} = w_I$ and the input bit is $\langle 0, 1\rangle$ (as seen in the most significant bit of the example), Algorithm 2 output

$$\begin{array}{rl} w_t = w_B = & \langle\infty,\infty,\infty,\infty,\infty,\infty,\ \infty,\\ & 1,\ \infty,\ 1,\ 1,\ 1,\ \infty,\ 1,\\ & 1,\ \infty,\ 1,\ 0,\ 1,\ \infty,\ 1,\\ & 1,\ \infty,\ 1,\ 1,\ 1,\ \infty,\ 1,\\ & \infty,\ \infty,\infty,\infty,\infty,\infty,\ \infty,\\ & 1,\ \infty,\ 1,\ 1,\ 1,\ \infty,\ 1,\\ & \infty,\ \infty,\infty,\infty,\infty,\infty,\infty\rangle,\end{array}$$

as shown in Table 1. $w_B \in Q_A$, and $(w_I, \langle 0, 1\rangle, w_B) \in \sigma_A$. Similarly, if the input is w_B and $\langle 1, 1\rangle$ as seen in bit 1 of the example, the algorithm input

$$\begin{array}{rl} w_C = & \langle 1,\ 1,\ 0,\ 1,\ 1,\ 1,\ \infty,\\ & 2,\ 1,\ 1,\ 1,\ 2,\ 1,\ \infty,\\ & 1,\ 2,\ 1,\ 2,\ 1,\ 2,\ \infty,\\ & 2,\ 1,\ 1,\ 1,\ 2,\ 1,\ \infty,\\ & \infty,\infty,\infty,\infty,\infty,\infty,\ \infty,\\ & 2,\ 2,\ 2,\ 2,\ 2,\ 2,\ \infty,\\ & 1,\ 2,\ 1,\ 2,\ 1,\ 2,\ \infty\rangle,\end{array}$$

as shown in Table 2. Also, $w_C \in Q_A$, $(w_B, \langle 1, 1\rangle, w_C) \in \sigma_A$.

Example 4. Construct the Markov chain $A = (Q_A, \Sigma, \sigma_A, I_A, P_A)$ for finding $AJW(E_m\{\{0, \pm 1\}, 1\})$.

– As $Ds = \{0, \pm 1\}$, $Cs = \{0, \pm 1\}$. Then,

$$w = \langle w_{\langle -1\rangle}, w_{\langle 0\rangle}, w_{\langle 1\rangle}\rangle.$$

The initial value of w, w_I is

$$w_I = \langle\infty, 0, \infty\rangle.$$

– Consider the loop in Lines 5-17. On the first iteration, $w_x = w_I$ in Line 7. If R is assigned to $\langle 0\rangle$ in Line 8, the result of the function MW in Line 9, w_y is

$$w_A = \langle 1, 0, 1\rangle.$$

Then, we add $\alpha = \langle w_I, \langle 0 \rangle, w_A \rangle$ to the set σ_A as shown in Line 10. The probability of the transition α is $\frac{1}{|\Sigma|} = \frac{1}{|\{0,1\}|} = \frac{1}{2}$. Also, we add w_A to the set Qu.

- Similarly, if $R = \langle 1 \rangle$, w_y is
$$w_B = \langle 0, 1, \infty \rangle.$$
Then, $w_B \in Q_A$, and $\langle w_I, \langle 1 \rangle, w_B \rangle \in \sigma_A$.
- Next, we explore the state w_A, as we explore the set Q_A by the breadth-first search algorithm. If $R = \langle 0 \rangle$, w_y is $\langle 1, 0, 1 \rangle$. And if $R = \langle 1 \rangle$, w_y is $\langle 1, 1, 0 \rangle$. Then,
$$\langle \langle 1, 0, 1 \rangle, \langle 0 \rangle, \langle 1, 0, 1 \rangle \rangle \in \sigma_A,$$
$$\langle \langle 1, 0, 1 \rangle, \langle 1 \rangle, \langle 0, 1, 1 \rangle \rangle \in \sigma_A.$$
The first transition is the self-loop. Hence, we need not to explore it again.
- We explore the state w_B, the result is
$$\langle \langle 0, 1, \infty \rangle, \langle 0 \rangle, \langle 1, 1, 2 \rangle \rangle \in \sigma_A,$$
$$\langle \langle 0, 1, \infty \rangle, \langle 1 \rangle, \langle 1, 2, \infty \rangle \rangle \in \sigma_A.$$
We note that $\langle 1, 1, 2 \rangle$ is equivalent to $\langle 0, 0, 1 \rangle$, and we denote it as $\langle 0, 0, 1 \rangle$. Also, $\langle 1, 2, \infty \rangle$ is equivalent to $\langle 0, 1, \infty \rangle$. Then, the second transition is the self-loop.
- Then, we explore the state $\langle 0, 1, 1 \rangle$. We get the condition
$$\langle \langle 0, 1, 1 \rangle, \langle 0 \rangle, \langle 1, 1, 2 \rangle \rangle \in \sigma_A,$$
$$\langle \langle 0, 1, 1 \rangle, \langle 1 \rangle, \langle 1, 2, 1 \rangle \rangle \in \sigma_A.$$
We denote $\langle 1, 1, 2 \rangle$ and $\langle 1, 2, 1 \rangle$ by $\langle 0, 0, 1 \rangle$, $\langle 0, 1, 0 \rangle$ respectively.
- We explore $\langle 0, 0, 1 \rangle$ and get the condition
$$\langle \langle 0, 0, 1 \rangle, \langle 0 \rangle, \langle 1, 0, 1 \rangle \rangle \in \sigma_A,$$
$$\langle \langle 0, 0, 1 \rangle, \langle 1 \rangle, \langle 0, 1, 1 \rangle \rangle \in \sigma_A.$$
- Exploring $\langle 0, 1, 0 \rangle$ makes we get
$$\langle \langle 0, 1, 0 \rangle, \langle 0 \rangle, \langle 1, 1, 1 \rangle \rangle \in \sigma_A,$$
$$\langle \langle 0, 1, 0 \rangle, \langle 1 \rangle, \langle 1, 1, 0 \rangle \rangle \in \sigma_A.$$
We denote $\langle 1, 1, 1 \rangle$ as $\langle 0, 0, 0 \rangle$.
- From the state $\langle 0, 0, 0 \rangle$, we get
$$\langle \langle 0, 0, 0 \rangle, \langle 0 \rangle, \langle 1, 0, 1 \rangle \rangle \in \sigma_A,$$
$$\langle \langle 0, 0, 0 \rangle, \langle 1 \rangle, \langle 0, 1, 0 \rangle \rangle \in \sigma_A.$$

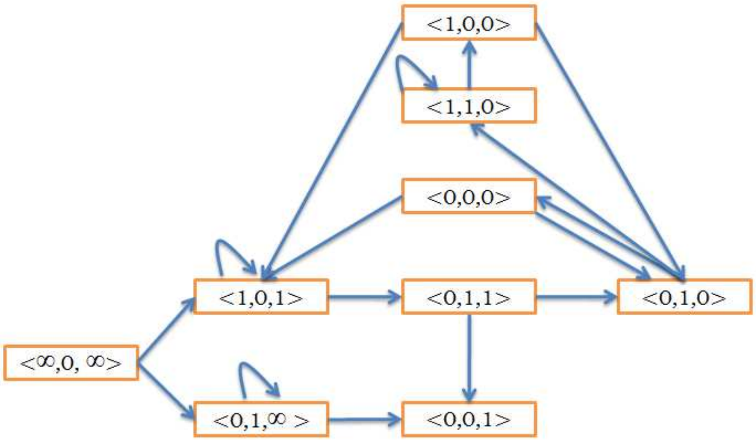

Fig. 1. The Markov chain constructed by Algorithm 3 used for finding $AJW(E_m\{\{0, \pm 1\}, 1\})$

- From the state $\langle 1, 1, 0\rangle$, we get

$$\langle\langle 1, 1, 0\rangle, \langle 0\rangle, \langle 2, 1, 1\rangle\rangle \in \sigma_A,$$

$$\langle\langle 1, 1, 0\rangle, \langle 1\rangle, \langle 1, 1, 0\rangle\rangle \in \sigma_A.$$

  We denote $\langle 2, 1, 1\rangle$ as $\langle 1, 0, 0\rangle$.
- Last, from the state $\langle 1, 0, 0\rangle$, we get

$$\langle\langle 1, 0, 0\rangle, \langle 0\rangle, \langle 1, 0, 1\rangle\rangle \in \sigma_A,$$

$$\langle\langle 1, 0, 0\rangle, \langle 1\rangle, \langle 0, 1, 0\rangle\rangle \in \sigma_A.$$

- We show the Markov chain in Figure 2.

Example 5. Construct the Markov chain $A = (Q_A, \Sigma, \sigma_A, I_A, P_A)$ for finding $AJW(E_m\{\{0, \pm 1\}, 2\})$.

- As $Ds = \{0, \pm 1\}$, $Cs = \{0, \pm 1\}$. Then,

$$\begin{aligned} w = \langle & w_{\langle -1,-1\rangle}, w_{\langle -1,0\rangle}, w_{\langle -1,1\rangle}, \\ & w_{\langle 0,-1\rangle}, \quad w_{\langle 0,0\rangle}, \quad w_{\langle 0,1\rangle}, \\ & w_{\langle 1,-1\rangle}, \quad w_{\langle 1,0\rangle}, \quad w_{\langle 1,1\rangle}\rangle. \end{aligned}$$

  The initial value of w, w_I is

$$\begin{aligned} w_I = \langle & \infty, \infty, \infty, \\ & \infty, \ 0, \ \infty, \\ & \infty, \infty, \infty\rangle. \end{aligned}$$

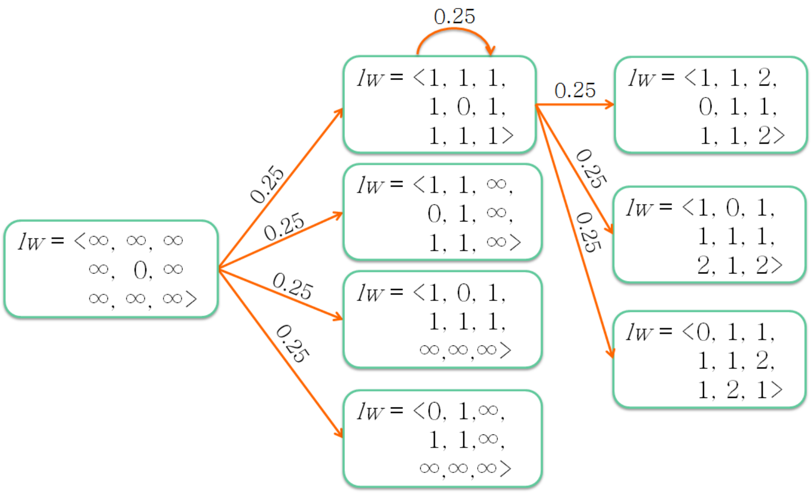

Fig. 2. The Markov chain constructed by Algorithm 3 after the second iteration of the loop in Lines 8-17

– Consider the loop in Lines 5-17. On the first iteration, $w_x = w_I$ in Line 7. If R is assigned to $\langle 0, 0\rangle$ in Line 8, the result of the function MW in Line 9, w_y is

$$\begin{aligned} w_y = w_A = \langle 1,\ 1,\ 1, \\ 1,\ 0,\ 1, \\ 1,\ 1,\ 1\rangle. \end{aligned}$$

Then, we add $\alpha = \langle w_I, \langle 0,0\rangle, w_A\rangle$ to the set σ_A as shown in Line 10. The probability of the transition α is $\frac{1}{|\Sigma\|} = \frac{1}{|\{0,1\}^2|} = \frac{1}{4}$. Also, we add w_A to the set Qu.
– The algorithm explores all $R \in \{0,1\}^2$. The result is shown in Figure 3.
– On the second iteration, $w_x = w_A$. If R is assigned to $\langle 0,0\rangle$, the result of the function MW is w_A itself. Therefore, the Markov chain consists of the self-loop at the state corresponding to w_A.

Let C be a number of states. We number each state $d \in Q_A$ as d_p where $1 \leq p \leq C$. Let $\pi^T = (\pi_i^T)$ be a probabilistic distribution at time T, i.e. π_i^T is the possibility that we are on state d_p after received input length T. Let $P = (P_{pq}) \in \mathbb{R}^{|Q_A| \times |Q_A|}$ be the transition matrix such that

$$P_{pq} = \sum_{R \in \Sigma} P_A(d_p, R, d_q).$$

Without loss of generality, assume d_1 representing the state that corresponds to the equivalence class of w_I. Then, $\pi^0 = (1, 0, \ldots, 0)^t$. From the equation $\pi^{T+1} = \pi^T P$, we find the stationary distribution such that $\pi^{T+1} = \pi^T$ by the eigen decomposition. In Appendix C, we prove that the stationary distribution always exists for any finite Markov chain generated from Algorithm 3.

The next step is to find the average weight from the stationary distribution π. Define WK as a function from σ_A to the set of integer by

$$WK(\tau) = w_{y,\langle \mathbf{0} \rangle} - w_{x,\langle \mathbf{0} \rangle},$$

when $\tau = (w_x, G, w_y) \in \sigma_A$. The function can be described as the change of the Hamming weight in the case that the carry tuple is $\langle \mathbf{0} \rangle$. We compute the average Hamming weight by the average value of the change in the Hamming weight when n is increased by 1 in the stationary distribution formalized as

$$AJW(E_m\{Ds, d\}) = \sum_{\tau \in \sigma_A} \frac{\pi_{f(\tau)} WK(\tau)}{|\Sigma|},$$

when $f(\tau) = w_x$ if $\tau = (w_x, G, w_y)$.

4.2 Analysis Results

By using the analysis method proposed in Subsection 4.1, we can find many interesting results on the average joint Hamming weight. We show some interesting results in Table 4. Our results match many existing result [1, 4, 9, 7]. And, we discover some results that have not been found in the literatures. We can describe the results as follows:

- When $d = 1$, we can find the average joint Hamming weight of all digit sets $Ds = \{0, \pm 1, \pm 3, \ldots, \pm(2h+1)\}$ when $h \leq 31$. If $h = 2^p - 1$ for some $p \in \mathbb{Z}$, our results match the existing results by Muir and Stinson [4]. And, we observe from the result that there is a relation between h and the average joint hamming weight. Let p be an integer such that

$$2^{p-1} - 1 < h < 2^p - 1,$$

$$AJW(\{0, \pm 1, \pm 3, \ldots, \pm(2h+1)\}, 1) = \frac{2^p}{(p+1)2^p + (h+1)}.$$

- When $d = 2$, we can find the average joint Hamming weight of $Ds = \{0, \pm 1, \pm 3, \ldots, \pm(2h+1)\}$ when $h \leq 5$. And, when $d = 3$, we can find the average joint hamming weight of $Ds = \{0, \pm 1, \pm 3\}$. The most interesting results is the case when $d = 2$, and $Ds = \{0, \pm 1, \pm 3\}$. This problem open by Solinas [1], and there are many works proposed the upper bound of the minimal average joint Hamming weight in this case. We can find the minimal average weight, and close this problem. We show our result compared with the previous works in Table 5.

Table 4. The average joint Hamming weight, $AJW(E_m\{Ds, d\})$, when $Ds = \{0, \pm1, \pm3, \ldots, \pm(2h+1)\}$ found by our analysis method, with the number of states in the Markov chain on each case

h / d	1	2	3	4
0	$\frac{1}{3} \approx 0.3333$ (Existing work [9]) (9 states)	$\frac{1}{2} = 0.5$ (Existing work [1]) (64 states)	$\frac{23}{39} \approx 0.5897$ (Existing work [7]) (941 states)	$\frac{115}{179} \approx 0.6424$ (Existing work [7]) (16782 states)
1	$\frac{1}{4} = 0.25$ (Existing work [4]) (38 states)	$\frac{281}{786} \approx 0.3575$ (Improved result) (3189 states)	$\frac{20372513}{49809043} \approx 0.4090$ (New result) (1216376 states)	
2	$\frac{2}{9} \approx 0.2222$ (New result) (70 states)	$\frac{1496396}{4826995} \approx 0.3100$ (New result) (19310 states)		
3	$\frac{1}{5} = 0.2$ (Existing work [4]) (119 states)	0.2660 (New result) (121601 states)		
4	$\frac{4}{21} \approx 0.1904$ (New result) (160 states)	0.2574 (New result) (130262 states)		
5	$\frac{2}{11} \approx 0.1818$ (New result) (207 states)	0.2342 (New result) (525620 states)		

Table 5. Comparing our result with the other preliminary researches when expand a pair of integers using $\{0, \pm1, \pm3\}$

Research	Average Joint Hamming Weight
Avanzi, 2002 [10]	$\frac{3}{8} = 0.3750$
Kuang et al., 2004 [11]	$\frac{121}{326} \approx 0.3712$
Moller, 2004 [12]	$\frac{4}{11} \approx 0.3636$
Dahmen et al., 2007 [5]	$\frac{239}{661} \approx 0.3616$
Our Result	$\frac{281}{786} \approx 0.3575$[Optimal]

5 Conclusion

In this paper, we propose the generalized minimal weight conversion algorithm for d integers. The algorithm can be applied to any finite digit set Ds. Then, we propose the algorithm to construct a Markov chain which can be used for finding the average joint Hamming weight automatically. As a result, we can discover some minimal average joint Hamming weights automatically without the prior knowledge of the structure of the digit set. This helps us able to explore the average weight of the unstructured set. For example, we find that the minimal average weight is $\frac{281}{786} \approx 0.3575$ when $d = 2$ and $Ds = \{0, \pm1, \pm3\}$. This improves the upper bound presented by Dahmen et al., that is $\frac{239}{661} \approx 0.3616$.

However, there are some gaps to improve this work as follows:

- Compared to the algorithm for each specific digit set, the generalized algorithm proposed in this paper is slower and consumes more memory. However, the efficient minimal weight conversion algorithm on the digit set that have never been explored might be able to derive from the algorithm.
- The number of the states in the Markov chain produced by Algorithm 3 is comparatively large. More efficient algorithm to generate the Markov chain will help us to be able to explore the average joint Hamming weight for the larger set.
- Our method is able to apply on the double-based number system [13, 14]. We have successfully proposing the minimal weight conversion for any d and Ds on this number system. However, we cannot find the fast analysis method from that conversion. In this state, we can improve the upper bound for the minimal average weight when $d = 2$ and $Ds = \{0, \pm1\}$ from 0.3945 [15] to 0.3883, but we still cannot find the exact value.

References

1. Solinas, J.A.: Low-weight binary representation for pairs of integers. Centre for Applied Cryptographic Research, University of Waterloo, Combinatorics and Optimization Research Report CORR (2001)
2. Heuberger, C., Muir, J.A.: Minimal weight and colexicographically minimal integer representation. Journal of Mathematical Cryptology **1** (2007) 297–328
3. Heuberger, C., Muir, J.A.: Unbalanced digit sets and the closest choice strategy for minimal weight integer representations. Designs, Codes and Cryptography **52**(2) (August 2009) 185–208
4. Muir, J.A., Stinson, D.R.: New minimal weight representation for left-to-right window methods. Department of Combinatorics and Optimization, School of Computer Science, University of Waterloo (2004)
5. Dahmen, E., Okeya, K., Takagi, T.: A new upper bound for the minimal density of joint representations in elliptic curve cryptosystems. IEICE Trans. Fundamentals **E90-A**(5) (May 2007) 952–959
6. Dahmen, E., Okeya, K., Takagi, T.: An advanced method for joint scalar multiplications on memory constraint devices. LNCS **3813** (2005) 189–204
7. Dahmen, E.: Efficient algorithms for multi-scalar multiplications. Diploma Thesis, Department of Mathematics, Technical University of Darmstadt (November 2005)

8. Okeya, K.: Joint sparse forms with twelve precomputed points. Technical Report of IEICE. ISEC **109**(42) (May 2009) 43–50 In Japanese.
9. Egecioglu, O., Koc, C.K.: Exponentiation using canonical recoding. Theoretical Computer Science **129** (1994) 407–417
10. Avanzi, R.: On multi-exponentiation in cryptography. Cryptology ePrint Archive **154** (2002)
11. Kuang, B., Zhu, Y., Zhang, Y.: An improved algorithm for $uP + vQ$ using JSF_3^1. LNCS **3089** (2004) 467–478
12. Moller, B.: Fractional windows revisited: Improved signed-digit representations for efficient exponentiation. LNCS **3506** (2005) 137–153
13. Dimitrov, V., Cooklev, T.V.: Two algorithms for modular exponentiation based on nonstandard arithmetics. IEICE Trans. Fundamentals **E78-A**(1) (January 1995) 82–87 special issue on cryptography and information security.
14. Dimitrov, V., Imbert, L., Mishra, P.K.: Efficient and secure elliptic curve point multiplication using double-base chains. In: Proc. of ASIACRYPT 2005. (2005) 59–78
15. Adikari, J., Dimitrov, V.S., Imbert, L.: Hybrid binary-ternary number system for elliptic curve cryptosystems. In: Proc. of EUROCRYPT 2009. (2009) 76–83
16. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. on Information Theory **IT-31** (1985) 469–472
17. Haggstrom, O.: Finite Markov Chains and Algorithmic Application. 1 edn. Volume 52 of London Mathematical Society, Student Texts. Cambride University, Coventry, United Kingdom (2002)
18. Schmidt, V.: Markov Chains and Monte-Carlo Simulation. Department of Stochastics, University Ulm (July 2006)
19. Suppakitpaisarn, V.: Optimal average joint hamming weight and digit set expansion on integer pairs. Master's thesis, The University of Tokyo (2009)
20. Suppakitpaisarn, V., Edahiro, M.: Fast scalar-point multiplication using enlarged digit set on integer pairs. Proc. of SCIS 2009 (2009) 14

Appendix A: The Carry Set

In this section, we present the algorithm to find the carry set Cs in Algorithm 1,2. We show the method in Algorithm 4. It is based on breadth-first search scheme. And, we find the upper bound of the cardinality of the carry set in Lemma 1.

Lemma 1. *Given the finite digit set Ds, Algorithm 3 always terminates. And,*

$$||Cs|| \leq \max Ds - \min Ds + 2,$$

when Cs is the output carry set.

Proof. Since

$$Cs = \{\frac{c-d}{2} \in \mathbb{Z} | d \in Ds \wedge c \in Cs\} \cup \{\frac{c-d+1}{2} \in \mathbb{Z} | d \in Ds \wedge c \in Cs\},$$

$$\min Cs \geq \frac{\min Cs - \max Ds}{2}.$$

Algorithm 4 Find the carry set of the given digit set

Require: the digit set Ds
Ensure: the carry set Cs

$Ct \leftarrow \{0\}$, $Cs \leftarrow \oslash$
while $Ct \neq \oslash$ **do**
  let $x \in Ct$
  $Ct \leftarrow Ct \cup (\{\frac{x+d}{2} \in \mathbb{Z} | d \in Ds\} - Cs - \{x\})$
  $Ct \leftarrow Ct \cup (\{\frac{x+d+1}{2} \in \mathbb{Z} | d \in Ds\} - Cs - \{x\})$
  $Cs \leftarrow Cs \cup \{x\}$
  $Ct \leftarrow Ct - \{x\}$
end while

Then,

$$\min Cs \geq -\max Ds.$$

Also,

$$\max Cs \leq -\min Ds + 1.$$

We conclude that if Ds is finite, Cs is also finite. And, Algorithm 3 always terminates.

$$||Cs|| \leq \max Ds - \min Ds + 2.$$

□

Appendix B: The Optimality of Algorithm 1,2

In this subsection, we present the mathematical proof that Algorithm 1,2 proposed in Section 3 is the minimal weight conversion.

Lemma 2. *For any positive integer $0 \leq t \leq n-1$. $Q_{t+1,\langle i,G_{t+1}\rangle}$, which are assigned in Line 7 of Algorithm 1, represent the minimal weight expansion of the prefix string length $n-t$ of the bit string $E_b\{d\}(r_1,\ldots,r_d)$, when the carry from less significant bits to the prefix is G. And, $w_{t+1,G_{t+1}}$ is the joint hamming weight of $Q_{t+1,\langle 1,G_{t+1}\rangle},\ldots,Q_{t+1,\langle d,G_{t+1}\rangle}$.*

Proof. We use the mathematic induction for proving this lemma.

We begin the proof by the case when $t = n-1$. In this case, all $Q_{n-1,\langle i,G_{n-1}\rangle}$ have length $(n-(n-1)) = 1$. The subsolution $Q_{n-1,\langle i,G_{n-1}\rangle}$ should satisfy

$$Q_{n-1,\langle i,G_{n-1}\rangle} = \langle ae_i \rangle,$$

if $AE \in Ds^d$, because it does not produce any carries to more significant bits. Then, $w_{n-1,G_{n-1}} = 0$ when $AE = \langle \mathbf{0} \rangle$ and $w_{n-1,G_{n-1}} = 1$ otherwise.

We initialize lw in Algorithm 1 Line 2 such that $w_{n,G_n} = 0$ if $G = \langle \mathbf{0} \rangle$, and $w_{n,G_n} = \infty$ otherwise. Then, $we_{R^*_{n-1}}$, which is assigned in Algorithm 2 Line 6, is ∞ if $G_n \neq \langle \mathbf{0} \rangle$. If there are some finite elements among we, $we_{R^*_{n-1}}$ will not be the minimal element on Algorithm 2 Line 11 and will not be assigned to $Q_{n-1,\langle i,G\rangle}$ in Algorithm 2 Line 15. Hence, all selected $EA = \langle ea_i \rangle_{i=1}^d$ satisfy

$$ce_i = \frac{ae_i - ea_i}{2} = 0,$$

for all $1 \leq i \leq d$. That means $ae_i = ea_i$, and we can conclude that $Q_{n-1,\langle i,G\rangle} = \langle ae_i \rangle$. Also, we prove that $w_{n-1,G_{n-1}} = 0$ when $G_{n-1} = \langle \mathbf{0} \rangle$ and $w_{n-1,G_{n-1}} = 1$ otherwise by Algorithm 2. We prove the statement when $T = n - 1$.

It is left to show that if the lemma holds when $t = K$, it also holds when $t = K - 1$, for any $K \geq 1$.

Assume that when $t = K$, $w_{K+1,G_{K+1}}$, $Q_{K+1,G_{K+1}}$ are the optimal weight and the optimal expansion of the prefix string length $n - K$ for any $G \in Cs^d$. We claim that w_{K,G_K}, Q_{K,G_K} are also the prefix string length $n - K + 1$.

First, we prove that w_{K,G_K} is the joint Hamming weight of

$$Q_{K,\langle 1,G_K\rangle}, \ldots, Q_{K,\langle d,G_K\rangle}$$

for any $G_K \in Cs^d$. It is obvious that we_{EA} selected in Algorithm 2 Line 11 equals $w_{K+1,CE}$, when $EA = \langle \mathbf{0} \rangle$ and $w_{K+1,CE} + 1$ otherwise, by Algorithm 2 Line 6 (CE is defined in Algorithm 2 Line 14). By the assignment in Algorithm 2 Line 15,

$$Q_{K,\langle i,G_K\rangle} = \langle Q_{K+1,\langle i,CE\rangle}, ea_i \rangle.$$

Since, the joint hamming weight of $Q_{K+1,\langle 1,CE\rangle}, \ldots, Q_{K+1,\langle d,CE\rangle}$ is equal to $w_{K+1,CE}$ by induction, the property also holds for each Q_{K,G_K}.

Next, we prove the optimality of $Q_{K,\langle i,G_K\rangle}$. Assume contradiction that there are some string $P_{K,\langle i,G_K\rangle}$ such that

$$P_{K,\langle i,G_K\rangle} \neq Q_{K,\langle i,G_K\rangle}$$

for some $1 \leq i \leq d$, and some $G_K \in Cs^d$. And, the joint hamming weight of $P_{K,\langle 1,G_K\rangle}, \ldots, P_{K,\langle d,G_K\rangle}$ is less than $Q_{K,\langle 1,G_K\rangle}, \ldots, Q_{K,\langle d,G_K\rangle}$. Let the last digit of $P_{K,\langle i,G_K\rangle}$ be lp_i. If $lp_i = ea_i$ for all $1 \leq i \leq d$, the carry is

$$\langle \frac{ae_i - ea_i}{2} \rangle_{i=1}^d = CE.$$

By induction, the joint Hamming weight $Q_{K+1,\langle 1,CE\rangle}, \ldots, Q_{K+1,\langle d,CE\rangle}$ is the minimal joint Hamming weight. Then, the joint hamming weight of P is greater or equal to Q. If $lp_i \neq ea_i$ for some $1 \leq i \leq d$, the carry is

$$H = \langle h_i \rangle_{i=1}^d = \langle \frac{ae_i - lp_i}{2} \rangle_{i=1}^d.$$

By induction, $Q_{K+1,\langle i,H\rangle}$ is the minimal weight expansion. Then,

$$JW(P_{K,\langle 1,H\rangle}, \ldots, P_{K,\langle d,H\rangle}) \geq JW(Q_{K+1,\langle 1,H\rangle}, \ldots, Q_{K+1,\langle d,H\rangle}) + JW(\langle lp_1 \rangle, \ldots, \langle lp_d \rangle),$$

when JW is the joint hamming weight function.

By the definition of WE, it is clear that

$$JW(Q_{K+1,\langle 1,H\rangle},\ldots,Q_{K+1,\langle d,H\rangle}) + JW(\langle lp_1\rangle,\ldots,\langle lp_d\rangle) = we_I,$$

when $I = \langle lp_1,\ldots,lp_d\rangle$.

In Algorithm 2 Line 11, we select the minimal value of we_{EA}. That is

$$we_{EA} \leq we_I.$$

As

$$we_{EA} = JW(Q_{K,\langle 1,G_K\rangle},\ldots,Q_{K,\langle d,G_K\rangle}),$$

we can conclude that

$$JW(P_{K,\langle 1,G_K\rangle},\ldots,P_{K,\langle d,G_K\rangle}) \geq JW(Q_{K,\langle 1,G\rangle},\ldots,Q_{K,\langle d,G\rangle}).$$

This contradicts our assumption. □

Theorem 1. *Let $Z = \langle \mathbf{0}\rangle$. $\langle Q_{0,\langle i,Z\rangle}\rangle_{i=1}^{d}$ in Algorithm 1 Line 9 is the minimal joint weight expansion of $r_1,\ldots,r_d$ on digit set Ds.*

Proof. $\langle Q_{0,\langle i,G\rangle}\rangle_{i=1}^{d}$ are the optimal binary expansion of the least significant bit by Lemma 2. Since there is no carry to the least significant bit, $\langle Q_{0,\langle i,\{\mathbf{0}\}\rangle}\rangle_{i=1}^{d}$ is the optimal solution. □

Appendix C: The Markov Chain from Algorithm 3

In this section, we prove that the set of states Q_A defined in Section 4 is finite. We also show that the Markov chain automatically generated by Algorithm 3 has a unique stationary distribution.

Lemma 3. *Let Λ be a largest integer of Ds, and $\{0,\pm 1,\pm \Lambda\} \subseteq Ds$. For any possible values of $lw = \langle lw\rangle_{G\in Cs^d}$, there exists $c \in \mathbb{Z}$ such that*

$$lw_{G_1} - lw_{G_2} < c,$$

for any $G_1, G_2 \in Cs^d$ such that lw_{G_1}, lw_{G_2} is a finite integer.

Proof. Let l be the possible value of lw, and $\alpha_1,\ldots,\alpha_d$ are the prefix string length β of the input that move the Markov chain to the state l. Let

$$\nu = \max(\max Cs, -\min Cs),$$

and $\gamma_1,\ldots,\gamma_d$ are the prefix string length $\beta - \nu$. They move the Markov chain to the state ll.

First, we show that for any $G \in Cs^d$

$$l_G \leq ll_{\langle \mathbf{0}\rangle} + \nu + d.$$

We consider the case that the last bit of α_ϵ is 0, for all $\epsilon \in \{1,\ldots,d\}$, and $G \in \{0,\pm 1\}^d$. Since $\{0,\pm 1\} \in Ds$,

$$l_G \leq ll_{\langle \mathbf{0} \rangle} + \mu \leq ll_{\langle \mathbf{0} \rangle} + \nu,$$

when μ is the joint Hamming weight of the substring of the input from the $(\beta - \nu + 1)^{th}$ significant bits to the β^{th} significant bit, $\lambda_1, \ldots, \lambda_d$.

Assume the last bit of α_ϵ is 1, for some $\epsilon \in \{1,\ldots,d\}$. If $G_\epsilon \in \{0,-1\}$, $ae_\epsilon \in Ds$. Next, we consider the case when $G_\epsilon = 1$, and $ae_\epsilon = 2$. If $2 \in Ds$, we prove the case. If not, we need to carry something to more significant bit. That carry can be dissolved if there exists 0 in some bit in λ_ϵ. If all number in λ_ϵ is 1, we might need to carry 1 into γ_ϵ. If there exists 0 in γ_ϵ, it might be changed to 1, that increase the joint Hamming weight at most 1. If not, the carry must be dissolved by changing some $c_1 \in Ds - \{0\}$ to $c_2 \in Ds - \{0\}$, that does not change the joint Hamming weight. Then, we prove the case.

Next, we prove the claim that

$$l_G \leq lu_H + 1,$$

when lu is the state of the Markov chain moved by the prefix string length $\beta - 1$ of the input, and H is defined as

$$H_\epsilon = \begin{cases} G_\epsilon & G_\epsilon \in \{0,\pm 1\} \\ \frac{G_\epsilon}{2} & |G_\epsilon| \geq 2 \text{ and } |G_\epsilon| \text{ is even} \\ \frac{G_\epsilon - 1}{2} & |G_\epsilon| \geq 2 \text{ and } |G_\epsilon| \text{ is odd and last digit of } \alpha_\epsilon \text{ is } 0 \\ \frac{G_\epsilon + 1}{2} & \text{otherwise .} \end{cases}$$

By the above equation,

$$|H_\epsilon| \leq |G_\epsilon| - 1 \text{ if } |G_\epsilon| \geq 2.$$

Then, by this claim

$$l_G \leq ll_H + \nu,$$

when $H \in \{0,\pm 1\}^d$. Hence,

$$l_G \leq ll_{\langle \mathbf{0} \rangle} + \nu + d.$$

In this paragraph, we show that the claim is correct. We consider G_ϵ as the carry to the bit, and H_ϵ as the carry to more significant bit. Let ω be the β^{th} significant bit of r_ϵ. In case that $2|G_\epsilon$, we can carry $\frac{G_\epsilon}{2}$ to more significant bits and leave this bit to ω. If $2|(G_\epsilon + 1)$ and ω is 0, we carry $\frac{G_\epsilon - 1}{2}$ to more significant bit, and leave 1. If the input is 1, we carry $\frac{G_\epsilon + 1}{2}$, and leave 0.

On the second part of the proof, we show the upper bound of $lw_{\langle \mathbf{0} \rangle}$. Let $\alpha_1, \ldots, \alpha_d$ be a string length β, and γ_i be a prefix of α_i length $\beta - \nu$, for some $\nu \in \mathbb{Z}$. α_i moves the Markov chain to the state l and γ_i moves the Markov chain to the state ll. We show that for any $G \in C^d$,

$$l_{\langle \mathbf{0} \rangle} \leq ll_G + \rho,$$

when ll is induced by the substring of the string which induce l. Refer to Algorithm 2, we select the best input in each output among Ds^d, by carrying $CA \in Cs^d$. But, we cannot output every possible Ds^d by the condition in Line 8. Let $D \subseteq Ds^d$ be a set of possible output, and $C \in Cs^d$ be a set of the possible carry from the bit. It is obvious that

$$l_{\langle \mathbf{0} \rangle} \leq ll_G + 1,$$

for any $G \in C$. Let $Cin \subseteq Cs$ be a set of possible carry to the input bit $I \in \{0, 1\}$ and $Cou \subseteq Cs$ is the possible carry produced by the bit and the set Cin. Define a function $\Delta : P(Cs) \times \{0, 1\} \rightarrow P(Cs)$ such that $\Delta(Ci, I) = Co$. We claim that if

$$\alpha_i \notin \{w | w \in (0^+1^+)^{|Cs|+1}\},$$

for some positive integer p, we can show that

$$Cs = \Delta \ldots \Delta(\Delta(\langle \mathbf{0} \rangle, \mathrm{I}_{\beta-|\mathrm{Cs}|}), \mathrm{I}_{\beta-|\mathrm{Cs}|+1}) \ldots, \mathrm{I}_{\beta}),$$

when I_t is the t^{th} significant bit of the string α_i. Let Λ be an odd number. It is clear that if $c \in Cin$ and $Cou = \Delta(Cin, I)$,

$$\{\frac{c+I+1}{2}, \frac{c+I-1}{2}, \frac{c+I+\Lambda}{2}, \frac{c+I-\Lambda}{2}\} \subseteq Cou,$$

when $c + I$ is odd, and

$$\{\frac{c+I}{2}\} \subseteq Cou,$$

when $c+I$ is even. We start with $Cin = \{0\}$. Since $\alpha_i \notin \{w | w \in (0^+1^+)^{|Cs|+1}\}$, there exist 1 in the bit string of α_i. And,

$$\Delta(\{0\}, 1) = \{0, 1, \frac{\Lambda+1}{2}, \frac{-\Lambda+1}{2}\}.$$

The set $\Delta(\{0\}, 1)$ contains both the odd and the even integer. Also, the set $\{\frac{c+I-\Lambda}{2}, \frac{c+I-1}{2}, \frac{c+I+1}{2}, \frac{c+I+\Lambda}{2}\}$, for any $c \in Cs$. If $c + I \neq \Lambda$, $\frac{c+I+\Lambda}{2} > c + I$. Then, the largest odd number of Cou is larger than the largest odd number of Cin. Also, the smallest odd number of Cou is smaller than the smallest odd number of Cou. Hence, if the largest or the smallest number is the odd number, we get the bigger set. As we have $\frac{c+I-1}{2}, \frac{c+I+1}{2} \in Cs$, the numbers lying between the smallest and the biggest number are filled with a small number of loop.

Last, we consider the case when α_i is the string in the regular language $(0^+1^+)^{|Cs|+1}$. Since $\{0, \pm 1\} \subseteq Ds$, we can represent α_i by NAF. The Hamming weight of representing the bit string α_i by NAF is at most $|Cs| + 2$.

□

Theorem 2. *The set Q_A is finite.*

Proof. Let $lwa = \langle lwa_G \rangle_{G \in Cs^d}$ be a possible value of lw such that

$$\alpha = \min_{G \in Cs^d} lwa_G.$$

Let $lwn = \langle lwn_G \rangle_{G \in Cs^d}$ such that

$$lwn_G = lwa_G - \alpha,$$

for any $G \in Cs^d$. It is obvious that lwn is equivalent to lwa, and they represent the same class in Q_A. The minimal value of lwn is 0, and the maximal value is c, as in Lemma 3. Then, there is $(c+1)^{||Cs||^d}$ possible classes of lw. And, the number of the members of the set Q_A is bounded by that number. □

Lemma 4. *For any possible values of* $lw = \langle lw \rangle_{G \in Cs^d}$*, there exists a sequence of members of* σ_A *such that*

$$\langle (lw, \langle \mathbf{0} \rangle, \mathrm{q}_1), (\mathrm{q}_1, \langle \mathbf{0} \rangle, \mathrm{q}_2), \ldots, (\mathrm{q}_\beta, \langle \mathbf{0} \rangle, \mathrm{lwc}) \rangle,$$

where $lw, q_1, \ldots, q_\beta, q_C \in Q_A$. $lwc = \langle lwc_G \rangle_{G \in Cs^d}$ *is the constant value such that*

$$lwc_G = JW(E_m\{Ds, d\}(g_1, \ldots, g_d)),$$

where $G = \langle g_i \rangle_{i=1}^d$.

Proof. For any possible values of lw, $lwa \in Q_A$. Let the inputs $r_1, \ldots, r_d$ be as follows:

– Let $n = max_i(\log_2(r_i))$.
– The first $n - \alpha$ bits of the inputs,

$$E_b n - 1, d(r_1, \ldots, r_d), \ldots, E_b \alpha, d(r_1, \ldots, r_d),$$

make the Markov chain be on that state lwa. As lwa is the possible value of lw, there always exists the bit string.
– The remaining of the bit string

$$E_b \alpha - 1, d(r_1, \ldots, r_d), \ldots, E_b 0, d(r_1, \ldots, r_d)$$

are $\langle \mathbf{0} \rangle$.

We claim that if α is large enough, the array lw is lwc when Algorithm 1 terminates.

First, we prove that

$$lwc_G = lwa_{\langle \mathbf{0} \rangle} + JW(\langle g_1 \rangle, \ldots, \langle g_d \rangle),$$

if $G \in Ds^d$. That means the carry to the α^{th} significant bit is $\langle \mathbf{0} \rangle$. We assume contradiction that there exists any carry from the least significant bit $K =$

$\langle k_i \rangle_{i=1}^{d}$ which make the joint weight lower. Let β be the joint Hamming weight of the zero-input part of the bit string,

$$lwa_{\langle \mathbf{0} \rangle} + JW(E_m\{Ds, d\}(g_1, \ldots, g_d) < lwa_L + \beta,$$

for some $L \in Cs^d$ that can be carry to the α^{th} when the carry from the least significant bit is K. It is obvious that $L \neq \langle \mathbf{0} \rangle$, as

$$\beta \geq JW(E_m\{Ds, d\}(g_1, \ldots, g_d).$$

Similaryly, it is also obvious that the carry between the bit number t and $t+1$ is not $\langle \mathbf{0} \rangle$.

Since Cs is a finite set, let

$$\max Cs < 2^{\gamma}.$$

And, let

$$\alpha > (c+1) \cdot \gamma + \epsilon,$$

when ϵ is the length of the shortest string that can represent

$$E_m\{Ds, d\}(g_1, \ldots, g_d).$$

Let the first $n - \alpha$ bits of $r_1, \ldots, r_d$ can be converted to the integer as $rf_1, \ldots, rf_d$. Carrying $L = \langle l_i \rangle_{i=1}^{d}$ makes us have to represent the number

$$rf_1 \cdot 2^{\alpha} + l_1, \ldots, rf_2 \cdot 2^{\alpha} + l_d$$

using the prefix length $n - 1$ of the solution. Since l_i is less than 2^{γ}, we cannot make the string that is all zeros between the bit number $\gamma + 1$ to 1 represent the value. This means the joint Hamming weight of the sub-solution between the bit number $\gamma + 1$ to 1 has to be more than 0.

Since the carry to the bit number α cannot be $\langle \mathbf{0} \rangle$, the carry from the bit number $\gamma+1$ is not $\langle \mathbf{0} \rangle$. As a result, the joint Hamming weight of the sub-solution between the bit number $2\gamma + 2$ to $\gamma + 2$ has to be more than 0. Similarly, the joint Hamming weight of the sub-solution between the bit number α to the bit number 1 is more than $c + 1$.

From Lemma 3,

$$lwa_L - lwa_{\langle \mathbf{0} \rangle} < c.$$

As $\beta \geq c + 1$, and $JW(E_m\{Ds, d\}(g_1, \ldots, g_d) \leq 1$, this contradicts our assumption that

$$lwa_{\langle \mathbf{0} \rangle} + JW(E_m\{Ds, d\}(g_1, \ldots, g_d) < lwa_L + \beta.$$

When, $G \notin Ds^d$, we start with the bit number ϵ, and use the same method to prove this lemma. □

Lemma 5. *The Markov chain constructed by Algorithm 3 always has one sink component.*

Proof. The component in the Markov chain the set $S \subseteq Q_A$ such that for all $p, q \in S$, there exists a path from p to q and from q to p.

The sink component is the component such that there are no edge from the component to the remaining part of the Markov chain.

From Lemma 4, we can conclude that there always exists a state that is reachable from every states in the Markov chain. And, all states that are reachable from that state form a component. The component is the sink component. If there exists a path from some nodes in the component, the inbound of the path should also be included in the component.

Since there always exists the path from all nodes to that sink component, it is impossible that the Markov chain has more than one sink components. □

Lemma 6. *Let* $lwc = \langle lwc_G \rangle_{G \in Cs^d}$ *such that*

$$lwc_G = JW(E_m\{Ds, d\}(g_1, \ldots, g_d)).$$

Then,

$$(lwc, \langle \mathbf{0} \rangle, \mathrm{lwc}) \in \sigma_{\mathrm{A}}.$$

Proof. In the proof of Lemma 4, we show that if

$$\alpha > (c+1) \cdot \gamma + \epsilon,$$

the Markov chain will be at the state lwc. We use the similar way to prove that this property also holds when

$$\alpha > (c+1) \cdot \gamma + \epsilon + 1.$$

This imply that

$$(lwc, \langle \mathbf{0} \rangle, \mathrm{lwc}) \in \sigma_{\mathrm{A}}.$$

□

Proposition 1. *(Corollary 2.5 of [18]) If the Markov chain is irreducible, all states have the same period.*

Proposition 2. *(Theorem 5.3 of [17]) Any irreducible and aperiodic Markov chain has exactly one stationary distribution.*

Theorem 3. *The Markov chain A generated by Algorithm 3 has exactly one stationary distribution.*

Proof. Let S be a sink component of the Markov chain A. Lemma 5 shows that S is unique. As the remaining part of the Markov chain $A - S$ has paths to the sink component with the positive probability, and there is no path from the sink component by the definition. It is obvious that in the stationary distribution of all states in $A - S$ is zero.

Then, it is left to prove that S has exactly one stationary distribution. By Proposition 2, we need to show that S is irreducible and aperiodic.

As S is a component, S is irreducible by the definition.

Let $s_i \in S$. Assume that start the Markov chain at s_i in time 0, and let the Markov chain come back to s_i in time $\{t_0, t_1, \ldots, t_k, \ldots\}$ with probability more than 0. The period of s_i , $pe(s_i)$ is defined as

$$pe(s_i) = \gcd(t_0, t_1, \ldots, t_k, \ldots).$$

The Markov chain is aperiodic if and only if all $s_i \in S$ have the period equals to 1.

By Lemma 6, there exists a node $s \in S$ that have a self loop. This node has the period equals to 1. And by Proposition 1, all states in S have the same period. Then, we can conclude that S is aperiodic. $\square$