Fast Elliptic Curve Cryptography Using Minimal Weight Conversion of d Integers

Vorapong Suppakitpaisarn[1] **Masato Edahiro**[2] **Hiroshi Imai**[1]

[1] Department of Computer Science, Graduate School of Information Science and Technology
The University of Tokyo,
Hongo, Bunkyo-ku, Tokyo, 113-8656
Email: mr_t_dtone@is.s.u-tokyo.ac.jp, imai@is.s.u-tokyo.ac.jp

[2] Department of Information Engineering, Graduate School of Information Science
Nagoya University,
Furo-cho, Chikusa-ku, Nagoya-shi, Aichi 464-8601
Email: eda@ertl.jp

Abstract

In this paper, we reduce computation time of elliptic curve signature verification scheme by proposing the minimal joint Hamming weight conversion for any binary expansions of d integers. The computation time of multi-scalar multiplication, the bottleneck operation of the scheme, strongly depends on the joint Hamming weight. As we represent the scalars using redundant representations, we may represent a number by many expansions. The minimal joint Hamming weight conversion is the algorithm to select the expansion which has the least joint Hamming weight. Many existing works introduce the conversions for some specific representations, and it is not trivial to generalize their algorithms to other representations. On the other hand, our conversion, based on the dynamic programming scheme, is applicable to find the optimal expansions on any binary representations. We also propose the algorithm to generate the Markov chain used for exploring the minimal average Hamming density automatically from our conversion algorithm. In general, the sets of states in our Markov chains are infinite. Then, we introduce a technique to reduce the number of Markov chain states to a finite set. With the technique, we find the average joint Hamming weight of many representations that have never been found. One of the most significant results is that, for the expansion of integer pairs when the digit set is $\{0, \pm 1, \pm 3\}$ often used in multi-scalar multiplication, we show that the minimal average joint Hamming density is 0.3575, which improves the upper bound value.

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

1 Introduction

The multi-scalar multiplication is the bottleneck operation of elliptic curve signature verification scheme. The operation is to compute

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

when r_i is a natural number, and P_i is a point on the elliptic curve. In this paper, we propose a method to reduce the computation time using a computer arithmetic technique considering the representation of each scalar r_i. In some redundant representations, we can represent each r_i in more than one way. Each way, called *expansion*, has a different value of Hamming weight, which directly affects the computation time of multi-scalar multiplications. Since the lower weight expansion makes the operation faster, many methods have been explored the lower weight expansion on many specific representations (1, 2, 3, 4, 5, 6, 7). These include the work by Solinas (1), which proposed the minimal joint weight expansion on an integer pair when digit set (defined in Section 2) is $\{0, \pm 1\}$. Also, the work by Heuberger and Muir (2, 3) presented the expansions for digit set $\{-l, -(l-1), \ldots, -1, 0, 1, \ldots, u-1, u\}$ for any natural number l, and positive integer u.

However, minimal weight conversions of many digit sets have not yet been found in the literature. This is caused by the fact that most of previous work presented the conversions based on the mathematical construction of the representation, which is hard to apply to many types of digit sets.

In this work, we propose a conversion method and an algorithm to find the average weight without concerning mathematical construction. This enables us to find the minimal weight conversions of digit sets used for multi-scalar multiplication. One of the significant result is the minimal weight conversion when the digit set is $\{0, \pm 1, \pm 3\}$ (8). Compared to the digit set that the minimal weight conversion have been found such as $\{0, \pm 1 \pm 2\}$ (2, 3), $\{0, \pm 1, \pm 3\}$ uses the same amount of memory to store the pre-computed points as $\{0, \pm 1, \pm 2\}$, but it is proved that $\{0, \pm 1, \pm 3\}$ has lower minimal average weight when $d = 2$.

To evaluate the effectiveness of each representation on elliptic curve cryptography, we utilize the average joint Hamming density, and we also propose a method to find the value for a class of digit set in this paper. Similar to the minimal weight conversions, most of the existing works proposed analysis based on the mathematical construction, which makes it hard to apply to many digit sets. On the other hand, we are able to calculate the value for our minimal weight con-

version algorithms, by proposing an algorithm to automatically generate the Markov chain from the conversion algorithms. In general, the sets of states in our Markov chains are infinite. Then, we introduce a technique to reduce the number of Markov chain states to a finite set.

One of our results 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 conversion methods and analysis for multi-scalar multiplication (5, 9, 10, 11). They can find the upper bound for the minimal average joint Hamming density. Our algorithm can find the minimal average joint Hamming density for this digit set, which is 0.3575. This improves the lowest upper bound 0.3616 in (5, 6).

It is shown in Appendix C that our minimal weight conversion algorithm is applicable to all finite digit sets. However, the algorithm to find average joint Hamming density is not. In many digit sets, the number of states in the Markov chain in the Markov chain is not finite, e.g. the representation in which $D_S = \{0, 1, 3\}$ and $d = 1$. In (12), we provide the proof of the finiteness of the Markov chain in a class of representation which cover all representations practically used in multi-scalar multiplication. Also, we are working on finding other reduction methods, which enable us to discover the value for wider class of representations.

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 a 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 density. Last, we conclude the paper in Section 5.

2 Definition

Let D_S be the digit set, n, d be positive integers, $E\{D_S, d\}$ be a conversion function from $\mathbb{Z}^d$ to $(D_S^n)^d$ such that if

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

when $\sum_{t=0}^{n-1} e_{i,t} 2^t = r_i$, where $r_i \in \mathbb{Z}$ and $e_{i,t} \in D_S$ for all $1 \le i \le d$. 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 $E\{D_S, d\}$. We also define a tuple of t-th bit of r_i as,

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

As a special case, let $E_b\{d\}$ be the binary conversion changing the integer to its binary representation where $D_S = \{0, 1\}$.

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

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

Also, define R_t as

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

. In our minimal weight conversion, R_t is considered as the input of bit t.

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

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

where

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

For instance,

$$JW_{E_b\{1\}}(12) = 2,$$

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

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

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

where

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

Since $K_t = O$, if

$$E\{D_S, d\}(r_1, \ldots, r_d)|_t = \langle \mathbf{0} \rangle,$$

we need not to perform point addition in that case. Thus, the number of point additions is $JW_{E\{D_S,d\}}(r_1, \ldots, r_d) - 1$. For instance, if

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

we can compute K as

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

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

When $\{0, 1\} \subset D_S$, we are able to represent some number $r_i \in \mathbb{Z}$ in more than one way. For instance, if $D_S = \{0, \pm 1\}$,

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

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

Let $E_m\{D_S, d\}$ be a minimal weight conversion where

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

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 $1 \le i \le d$,

$$\sum_{t=0}^{n-1} jw'_t \ge JW_{E_m\{D_S,d\}}(r_1, \ldots, r_d),$$

and

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

For instance,

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

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

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

If $D_{S_2} \subseteq D_{S_1}$, it is obvious that

$$JW_{E_m\{D_{S_2},d\}}(r_1, \ldots, r_d) \geq JW_{E_m\{D_{S_1},d\}}(r_1, \ldots, r_d).$$

Thus, we can increase the efficiency of the scalar-point multiplication by increaseing the size of D_S. However, the bigger D_S needs more precomputation tasks. If $d = 2$, we need one precomputed point when $D_S = \{0, 1\}$, but we need 10 precomputed points when $D_S = \{0, \pm 1, \pm 3\}$.

Then, one of the contributions of this paper is to evaluate an efficiency of each digit set D_S on multi-scalar multiplication. We use the average joint Hamming density defined as

$$AJW(E\{D_S, d\}) = \lim_{n\to\infty} \sum_{r_1=0}^{2^n-1} \cdots \sum_{r_d=0}^{2^n-1} \frac{JW_{E\{D_S,d\}}(r_1, \ldots, r_d)}{n2^{dn}}.$$

It is easy to see that $AJW(E_b\{d\}) = 1 - \frac{1}{2^d}$. In this paper, we find the value $AJW(E_m\{D_S, d\})$ of some D_S and d. Some of these values have been found in the literature 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 a minimal weight conversion algorithm based on the dynamic programming scheme. The input is $\langle r_1, \ldots, r_d \rangle$, and the output is $E_m\{D_S, d\}(r_1, \ldots, r_d)$, which is the minimal weight expansion of the input using the digit set D_S. The algorithm begins from the most significant bit (bit $n-1$), R_{n-1}, and processes left-to-right to the least significant bit (bit 0), R_0.

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

- The *carry array* $G_t = \langle g_{1,t}, \ldots, g_{d,t} \rangle$ is a possible integer array as carry from bit $t-1$. For the input
$$R_t = \langle e_{1,t}, \ldots, e_{d,t} \rangle$$
and output
$$R^*_t = \langle e^*_{1,t}, \ldots, e^*_{d,t} \rangle \in D^d_S,$$
the following formula should be satisfied:
$$R_t + G_t = R^*_t + 2G_{t+1}.$$
Since $R^*_t \in D^d_S$, possible values of $g_{i,t}$ is calculated from D_S. We define the carry set C_S by the set of possible carry values for D_S. In Appendix B, we give the detail of the carry set C_S, and prove that the set is always finite if D_S is finite. For example, when the digit set $D_S = \{0, \pm 1, \pm 3\}$, the carry set is $C_S = \{0, \pm 1, \pm 2, \pm 3\}$. It is noted that $G_t = \langle \mathbf{0} \rangle$ for $t = 0$ and $t = n$ as boundary conditions.

- The *minimal weight array* w_t is the array of the positive integer w_{t,G_t} for any $G_t \in C^d_S$. The integer w_{t,G_t} is the minimal joint weight of the first $n-t$ bits of the input $r_1, \ldots, r_d$ $(\lfloor \frac{r_1}{2^t} \rfloor, \ldots, \lfloor \frac{r_d}{2^t} \rfloor)$ for carry $G_t = \langle g_{i,t} \rangle^d_{i=1}$, e.g.
$$w_{t,G_t} = JW_{E_m\{D_S,d\}}(\lfloor \frac{r_1}{2^t} \rfloor + g_{1,t}, \ldots, \lfloor \frac{r_d}{2^t} \rfloor + g_{d,t}).$$

- 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 $G_t \in C^d_S$. 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 $G_t = \langle g_{i,t} \rangle^d_{i=1}$, e.g.
$$Q_{t,G_t} = \langle Q_{t,\langle i,G_t \rangle} \rangle^d_{i=1} = E_m\{D_S, d\}(\lfloor \frac{r_1}{2^t} \rfloor + g_{1,t}, \ldots, \lfloor \frac{r_d}{2^t} \rfloor + g_{d,t}).$$
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 $Q_{t,\langle 1,G_t \rangle}, \ldots, Q_{t,\langle d,G_t \rangle}$. There may exist some $g_{i,t} \in C_S$ such that $\lfloor \frac{r_1}{2^t} \rfloor + g_{i,t}$ can not be represented using the string length $n-t$ of D_S. In that case, we represent $Q_{t,\langle i,G_t \rangle}$ with the null string, and assign w_{t,G_t} to ∞.

In the process at the bit t, we find the minimal weight array w_t and the subsolution array Q_t from the input R_t, the minimal weight array w_{t+1}, and the subsolution array Q_{t+1}. For the process, we define the function MW such that

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

Since $w_t = \langle w_{t,G_t} \rangle_{G_t \in C^d_S}$ and $Q_t = \langle Q_{t,G_t} \rangle_{G_t \in C^d_S}$, 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, and we can use only two arrays to represent all w_t and w_{t+1} to reduce memory consumption. Similarly, we store all Q_t using two arrays.

Here, we will show the basic idea of our proposed algorithm with an example.

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

- *Step 1* Consider the most significant bit, the input
$$R_2 = E_b\{2\}(3, 7)|_{t=2} = \langle 0, 1 \rangle.$$

For the digit set $D_S = \{0, \pm 1, \pm 3\}$, the carry set is calculated as $C_S = \{0, \pm 1, \pm 2, \pm 3\}$. Thus, there are 25 pairs for possible carries G_2. For example, when $G_2 = \langle 0, -1\rangle$, $R_2 + G_2 = \langle 0, 1\rangle + \langle 0, -1\rangle = \langle 0, 0\rangle$, so that the Hamming weight $w_{2,\langle 0,-1\rangle} = 0$. As a boundary condition, we do not generate carry from the most significant bit because we want to keep the length of the bit string unchanged.

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_2} is 1 for any G, such that

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

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

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

and $w_{2,\langle 0,1\rangle} = \infty$, because 2 is not in D_S. The Hamming weight $w_{2,G}$ is ∞ for any G_2, such that $R_2 + G_2 \notin D_S^d$.

- *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 $G_{t+1} \in C_S^d$ is the carry to the most significant bit and $R_t^* \in D_S^d$ is the candidate for the output. That is

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

The Hamming weight should be

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

From the calculation for bit 2 shown in Page 3,

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

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

And,

$$\begin{aligned}JW(\langle 1, 0\rangle) &= JW(\langle 0, 3\rangle) \\ &= JW(\langle 0, -1\rangle) \\ &= JW(\langle 0, -3\rangle) \\ &= 1.\end{aligned}$$

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 1.

- *Step 3* On the least significant bit, the input $R_0 = \langle 1, 1\rangle$. Also, as a boundary condition, we set $G_0 = \langle \mathbf{0}\rangle$, and therefore, the value $w_{0,\langle 0,0\rangle}$ is the minimal Hamming weight. 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 $G_1 \in C_S^d$, $R_0^* \in D_S^d$. 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 2. Shown in the table, the minimal Hamming weight is

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

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

Require: $r_1, \ldots, r_d$
The desired digit set D_S
Ensure: $E_m\{D_S, d\}(r_1, \ldots, r_d)$
1: Let C_S be a carry set such that for all $c \in C_S$ and $d \in D_S$, $\frac{c+d}{2}, \frac{c+d+1}{2} \in C_S$.
2: Let w_t be an array of w_{t,G_t} for any $G_t \in C_S^d$.
$w_{n,G_n} \leftarrow 0$ if $G_{n-1} = \langle \mathbf{0}\rangle$.
$w_{n,G_n} \leftarrow \infty$ otherwise.
3: Let $Q_t \leftarrow \langle Q_{t,\langle i,G_t\rangle}\rangle$ for any $1 \le i \le d$ and $G_t \in C_S^d$.
All $Q_{n,\langle i,G_t\rangle}$ are initiated to a null string.
4: **for** $t \leftarrow n - 1$ to 0 **do**
5: $R_t \leftarrow E_b\{d\}(r_1, \ldots, r_d)|_t$.
6: $(w_t, Q_t) \leftarrow MW(w_{t+1}, Q_{t+1}, R_t)$
(We define the function MW in Algorithm 2)
7: **end for**
8: Let $Z \leftarrow \langle \mathbf{0}\rangle$.
$E_m\{D_S, d\}(r_1, \ldots, r_d) \leftarrow \langle Q_{0,\langle i,Z\rangle}\rangle_{i=1}^d$

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
1: **for all** $G_t = \langle g_{i,t}\rangle_{i=1}^d \in C_S^d$ **do**
2: $AE = \langle ae_i\rangle_{i=1}^d \leftarrow R_t + G_t$
3: **for all** $R_t^* = \langle r_{i,t}^*\rangle_{i=1}^d \in D_S^d$ **do**
4: **if** $2|(ae_i - r_{i,t}^*)$ for all $1 \le i \le d$ **then**
5: $G_{t+1} \leftarrow \langle \frac{ae_i - r_{i,t}^*}{2}\rangle_{i=1}^d$
6: $we_{R_t^*} \leftarrow w_{t+1,G_{t+1}}$ if $G_{t+1} = \langle \mathbf{0}\rangle$.
$we_{R_t^*} \leftarrow w_{t+1,G_{t+1}} + 1$ otherwise.
7: **else**
8: $we_{R_t^*} \leftarrow \infty$
9: **end if**
10: **end for**
11: Let we_{EA} is the one of the minimal values among we.
12: $w_{t,G_t} \leftarrow we_{EA}$
13: Let $EA = \langle ea_i\rangle_{i=1}^d$.
14: $CE = \langle ce_i\rangle_{i=1}^d \leftarrow \langle \frac{ae_i - ea_i}{2}\rangle_{i=1}^d$
15: $Q_{t,\langle i,G_t\rangle} \leftarrow \langle Q_{t+1,\langle i,CE\rangle}, ea_i\rangle$ for all $1 \le i \le d$
16: **end for**

We show the detailed algorithm in Algorithm 1 and Algorithm 2, which is shown on Page 5. There are some points to be noted as follows:

Table 1: 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 computed as in the first bullet of Example 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 2: 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 1), $JW(R_0^*)$, $w_{1,G_1} + JW(R_0^*)$ of each G_1,R_0^*

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

- Algorithms 1,2 have been proved to be the optimal algorithm, i.e. minimal joint Hamming weight conversion algorithm. The proof is shown in Appendix C.
- The size of the array w_t and Q_t is equal to $||C_S||^d$ and $dn||C_S||^d$ 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. It might be possible to make the array size lower when the method is implemented 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.

4 Average Joint Hamming Density Analysis with Markov Chain

In this section, we propose the algorithm to analyze the average joint Hamming density 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 w_x, w_y 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

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

4.1 Markov Chain Construction Algorithm

Algorithm 3 Construct the Markov chain used for finding the minimal average Hamming density

Require: the digit set D_S
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: C_S : carry set for D_S
3: $w_I \leftarrow \langle w_{I,G}\rangle_{G \in C_S^d}$, 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.

From Algorithms 1,2, we propose Algorithm 3 to construct the Markov chain. We illustrate the main idea of Algorithm 3 in Figure 1 for $D_S = \{0, \pm 1\}$ and $d = 1$. Thus, the figure shows the Markov chain for finding $AJW(E_m\{\{0, \pm 1\}, 1\})$. Initially, the Markov chain is considered as a tree rooted by the node $\langle \infty, 0, \infty\rangle$, which is the initial state of Algorithm 1. Note that $C_S = \{0, \pm 1\}$ for $D_S = \{0, \pm 1\}$, and each state of the Markov chain represents w which contains three values $\langle w_{\langle -1\rangle}, w_{\langle 0\rangle}, w_{\langle 1\rangle}\rangle$ associated with each value in C_S. Because of the boundary solution explained in Section 3, the initial state should be $\langle \infty, 0, \infty\rangle$. Each node w_x has two children, $w_y = MW(w_x, \langle 0\rangle)$ and $w_{y'} = MW(w_x, \langle 1\rangle)$. The Markov chain should be a tree with infinite length.

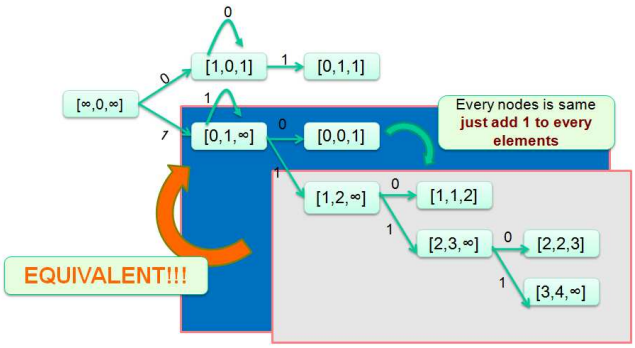

Figure 1: A state of constructing the Markov chain for finding $AJW(E_m\{\{0,\pm1\},1\})$ by Algorithm 3. States $\langle 0,1,\infty\rangle$ and $\langle 1,2,\infty\rangle$ are shown to be equivalent, and can be grouped.

Now, consider nodes with same label as one node. Also, all children of $\langle 1,2,\infty\rangle$ are almost similar to the children of $\langle 0,1,\infty\rangle$. The only difference is the addition of one to every entries of each node. Then, we can consider $\langle 0,1,\infty\rangle$ to be equivalent to $\langle 1,2,\infty\rangle$, and note this information as the weight of the transition from $\langle 0,1,\infty\rangle$ to $\langle 1,2,\infty\rangle$. We consider $\langle 0,1,\infty\rangle$ and $\langle 1,2,\infty\rangle$ to be in the same *equivalent class* . This example will be shown in detail in Example 3 of Appendix A.

Let

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

where

- Q_A is a set of states,
- Σ is the alphabet, i.e., the set of all possible digits,
- $\sigma_A \subseteq Q_A \times \Sigma \times Q_A$ is a set of transitions,
- $I_A : Q_A \to \mathbb{R}^+$ is an initial-state probabilities for each state in Q_A,
- $P_A : \sigma_A \to \mathbb{R}^+$ 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 Algorithms 1,2. Let $w_x = \langle w_{x,G}\rangle_{G\in C_S^d}$ and $w_{x'} = \langle w_{x',G}\rangle_{G\in C_S^d}$ 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 $G \in C_S^d$.

  With this method, the number of states in Markov chain becomes finite in our interested digit sets. However, the number can be very large when the digit set becomes larger. For example, the number of states is $1,216,376$ for $d = 3$ and $D_S = \{0, \pm1, \pm3\}$.

  In many digit sets, the number of states in the Markov chain in the Markov chain is not finite, e.g. the representation in which $D_S = \{0, 1, 3\}$ and $d = 1$. In (12), we provide the proof of the finiteness of the Markov chain in a class of representation which cover all representations practically used in multi-scalar multiplication. Also, we are working on finding other reduction methods, which enable us to discover the value for wider class of representations.

- To find the average joint Hamming density, we need to find the possibility that the Markov chain is on each equivalence class after we input a bit string length $n \to \infty$. 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_y and $w_{y'}$ 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

  $$\sigma_A = \{(w_x, R, w_y) | w_y = MW(w_x, R)\}.$$

- 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 possibility $P_A(\gamma) = \frac{1}{|\Sigma|}$ for all $\gamma \in \sigma_A$. This is shown in Algorithm 3 Line 11.

Let C be a number of states. We number each state $d \in Q_A$ as d_p where $1 \le p \le C$. Let $\pi^T = (\pi_p^T)$ be a probabilistic distribution at time T, i.e. π_p^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.

The next step is to find the average Hamming density 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 density by the average value of the change in the Hamming weight when n is increased by 1 in the stationary distribution formalized as

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

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

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

h	$d=1$	$d=2$	$d=3$	$d=4$
0	$\frac{1}{3} \approx 0.3333$ (Existing work (14)) (9 states)	$\frac{1}{2} = 0.5$ (Existing work (1)) (64 states)	$\frac{23}{39} \approx 0.5897$ (Existing work (15)) (941 states)	$\frac{115}{179} \approx 0.6424$ (Existing work (15)) (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	$\frac{2}{9} \approx 0.2222$ (Existing work (11)) (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	$\frac{4}{21} \approx 0.1904$ (Existing work (11)) (160 states)	0.2574 (New result) (130262 states)		

Table 3: Comparing our result with the other previous works when expand a pair of integers using $\{0, \pm1, \pm3\}$

Research	Average Joint Hamming Weight
Avanzi, 2002 (9)	$\frac{3}{8} = 0.3750$
Kuang et al., 2004 (10)	$\frac{121}{326} \approx 0.3712$
Moller, 2004 (11)	$\frac{4}{11} \approx 0.3636$
Dahmen et al., 2007 (5)	$\frac{239}{661} \approx 0.3616$
Our Result	$\frac{281}{786} \approx 0.3575$[Optimal]

4.2 Analysis Results

By using the analysis method proposed in Subsection 4.1, we can find many crucial results on the average joint Hamming density. Some results are shown in Table 4. Our results match many existing result (1, 4, 7, 14). And, we discover some results that have not been found in the literature. We can describe the results as follows:

- When $d = 1$, we can find the average joint Hamming density of all digit sets $D_S = \{0, \pm1, \pm3, \ldots, \pm(2h+1)\}$ 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 results that there is a relation between h and the average joint Hamming density. Let p be an integer such that

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

$$AJW(D_{S_h}, 1) = \frac{2^p}{(p+1)2^p + (h+1)}.$$

  where $D_{S_h} = \{0, \pm1, \pm3, \ldots, \pm(2h+1)\}$.
- When $d = 2$, we can find the average joint Hamming density of $D_S = \{0, \pm1, \pm3, \ldots, \pm(2h+1)\}$ when $h \leq 5$. And, when $d = 3$, we can find the average joint Hamming density of $D_S = \{0, \pm1, \pm3\}$. The most significant results is the case when $d = 2$, and $D_S = \{0, \pm1, \pm3\}$. This problem was raised as a future work by Solinas in 2001 (1), and there are many works proposed the upper bound of the minimal average joint Hamming density in this case. We can find the minimal average Hamming density, and give the solution of this open problem. We show our result compared with the previous works in Table 3.

5 Conclusion and Future Works

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

Many ideas proposed in this paper are also introduced in the minimal weight conversion algorithm for double-base chain (16), and the analysis of the efficiency of the chain is one of the most interesting problem we are challenging.

References

[1] Solinas, J.A.: Low-weight binary representation for pairs of integers. Combinatorics and optimization research report CORR, Centre for Ap-

plied Cryptographic Research, University of Waterloo (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) (2009) 185–208

[4] Muir, J.A., Stinson, D.R.: New minimal weight representation for left-to-right window methods. Technical report, 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) (2007) 952–959

[6] Dahmen, E., Okeya, K., Takagi, T.: An advanced method for joint scalar multiplications on memory constraint devices. In: Security and Privacy in Ad-hoc and Sensor Networks. Volume 3813/2005 of Lecture Notes in Computer Science., Springer (2005) 189–204

[7] Dahmen, E.: Efficient algorithms for multi-scalar multiplications. Master's thesis, Department of Mathematics, Technical University of Darmstadt (2005)

[8] Okeya, K.: Joint sparse forms with twelve precomputed points. IEICE Technical Report **IEICE-109**(IEICE-ISEC-42) (2009) 43–50

[9] Avanzi, R.: On multi-exponentiation in cryptography. Cryptology ePrint Archive, 2002/154 (2002)

[10] Kuang, B., Zhu, Y., Zhang, Y.: An improved algorithm for $uP + vQ$ using JSF_3^1. In: Applied Cryptography and Network Security. Volume 2004/3089 of Lecture Notes in Computer Science., Springer (2004) 467–478

[11] Moller, B.: Fractional windows revisited:improved signed-digit representations for efficient exponentiation. In: Information Security and Cryptology - ICISC 2004. Volume 3506/2005 of Lecture Notes in Computer Science., Springer (2005) 137–153

[12] Suppakitpaisarn, V., Edahiro, E., Imai, H.: Calculating Average Joint Hamming Weight for Minimal Weight Conversion of d Integers In: Workshop on Algorithms and Computation - WALCOM 2012. Lecture Notes in Computer Science., Springer (2012), (to be appeared)

[13] Haggstrom, O.: Finite Markov Chains and Algorithmic Application. 1 edn. Volume 52 of London Mathematical Society, Student Texts. Cambride University, Coventry, United Kingdom (2002)

[14] Egecioglu, O., Koc, C.K.: Exponentiation using canonical recoding. Theoretical Computer Science **129** (1994) 407–417

[15] Heuberger, C., Katti, R., Prodinger, H., Ruan, X.: The alternating greedy expansion and applications to left-to-right algorithms. IEICE Trans. Fundamentals **E90-A** (2007) 341–356

[16] Suppakitpaisarn, V., Edahiro, E., Imai, H.: Fast elliptic curve cryptography using optimal double-base chains Cryptology ePrint Archive, **Report 2011/030** (2011)

[17] Schmidt, V.: Markov Chains and Monte-Carlo Simulation. Department of Stochastics, University Ulm (2006)

Appendix A: More Examples

In this section, we give more examples for better understanding of the algorithm proposed in Section 3,4. Example 2 is the example for the minimal weight conversion in Section 3, and Examples 3,4 are the examples for the Markov chain construction proposed in Section 4.

Example 2 Compute $E_m\{\{0,\pm1,\pm3\},2\}(23,5)$ using Algorithm 1,2.

- $E_b\{2\}(23,5) = \langle(10111),(00101)\rangle$.
- When $D_S = \{0,\pm1,\pm3\}$, $Cs = \{0,\pm1,\pm2,\pm3\}$.
- To simplify the explanation, we present it when the loop in Algorithm 1 Lines 4-7 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, $w_t = \langle w_{t,G_t}\rangle_{G_t}$ where $G_t \in \{0,\pm1,\pm2,\pm3\}^2$. 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,

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

$$Q_{1,\langle 1,\langle 0,0\rangle\rangle} = (1011),$$
$$Q_{1,\langle 1,\langle 1,0\rangle\rangle} = (0300),$$
$$Q_{1,\langle 1,\langle 2,0\rangle\rangle} = (0301),$$
$$Q_{1,\langle 2,\langle 0,0\rangle\rangle} = (0010),$$
$$Q_{1,\langle 2,\langle 1,0\rangle\rangle} = (0010),$$
$$Q_{1,\langle 2,\langle 2,0\rangle\rangle} = (0010).$$

- 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.

Example 3 Construct the Markov chain $A = (Q_A, \Sigma, \sigma_A, I_A, P_A)$ for finding $AJW(E_m\{\{0, \pm 1\}, 1\})$.

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

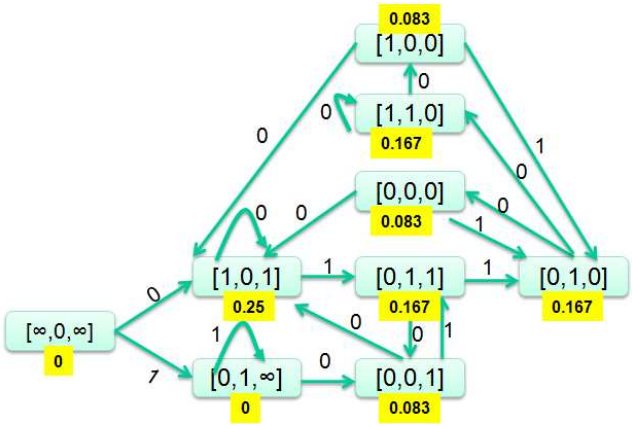

Figure 2: The Markov chain constructed by Algorithm 3 used for finding $AJW(E_m\{\{0, \pm 1\}, 1\})$

- 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

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

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

- 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 1.

Example 4 Construct the Markov chain $A = (Q_A, \Sigma, \sigma_A, I_A, P_A)$ for finding $AJW(E_m\{\{0, \pm 1\}, 2\})$.

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

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

The initial value of w, w_I is

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

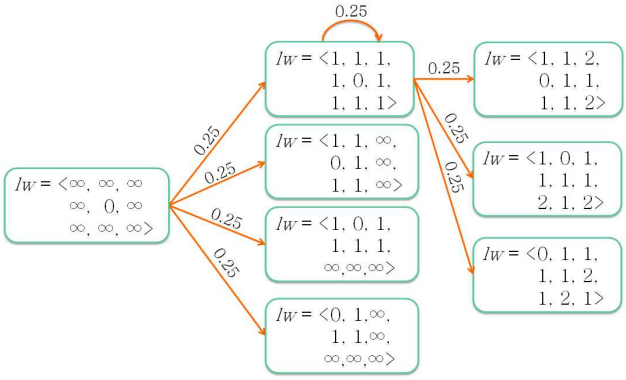

Figure 3: The Markov chain constructed by Algorithm 3 after the second iteration of the loop in Lines 8-17

- 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

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

  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 2.

- 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.

Appendix B: the Carry Set

In this section, we present the algorithm to find the carry set C_S in Algorithms 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 5.1.

Algorithm 4 Find the carry set of the given digit set

Require: the digit set D_S
Ensure: the carry set C_S
1: $Ct \leftarrow \{0\}$, $C_S \leftarrow \oslash$
2: **while** $Ct \neq \oslash$ **do**
3: let $x \in Ct$
4: $Ct \leftarrow Ct \cup (\{\frac{x+d}{2} \in \mathbb{Z} | d \in D_S\} - C_S - \{x\})$
5: $Ct \leftarrow Ct \cup (\{\frac{x+d+1}{2} \in \mathbb{Z} | d \in D_S\} - C_S - \{x\})$
6: $C_S \leftarrow C_S \cup \{x\}$
7: $Ct \leftarrow Ct - \{x\}$
8: **end while**

Lemma 5.1 *Given the finite digit set D_S, Algorithm 3 always terminates. And,*

$$||C_S|| \leq \max D_S - \min D_S + 2,$$

when C_S is the output carry set.

Proof Since

$$C_S = \{\frac{c-d+e}{2} \in \mathbb{Z} | d \in D_S \wedge c \in C_S \wedge e \in \{0,1\}\},$$

$$\min C_S \geq \frac{\min C_S - \max D_S}{2}.$$

Then,

$$\min C_S \geq -\max D_S.$$

Also,

$$\max C_S \leq -\min D_S + 1.$$

We conclude that if D_S is finite, C_S is also finite. And, Algorithm 3 always terminates.

$$||C_S|| \leq \max D_S - \min D_S + 2. \quad \blacksquare$$

Appendix C: The Optimality of Algorithm 1,2

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

Lemma 5.2 *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_{t+1}. 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,

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

when JW is the joint hamming weight function.

By the definition of WE, it is clear that

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

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

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

This contradicts our assumption. ∎

Theorem 5.3 *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 5.2. 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. ∎