ANALYSIS OF LINEAR COMBINATION ALGORITHMS IN CRYPTOGRAPHY

PETER J. GRABNER[†], CLEMENS HEUBERGER[‡], HELMUT PRODINGER[*], AND JÖRG M. THUSWALDNER[¶]

Abstract. Several cryptosystems rely on fast calculations of linear combinations in groups. One way to achieve this is to use joint signed binary digit expansions of small "weight." We study two algorithms, one based on non adjacent forms of the coefficients of the linear combination, the other based on a certain joint sparse form specifically adapted to this problem. Both methods are sped up using the sliding windows approach combined with precomputed lookup tables. We give explicit and asymptotic results for the number of group operations needed assuming uniform distribution of the coefficients. Expected values, variances and a central limit theorem are proved using generating functions.

Furthermore, we provide a new algorithm which calculates the digits of an optimal expansion of pairs of integers from left to right. This avoids storing the whole expansion, which is needed with the previously known right to left methods, and allows an online computation.

1. Introduction

In many public key cryptosystems, raising one or more elements of a given group to large powers plays an important role (cf. for instance [2, 8]). In practice, the underlying groups are often chosen to be the multiplicative group of a finite field $\mathbb{F}_q$ or the group law of an elliptic curve (elliptic curve cryptosystems).

Let P be an element of a given group, whose group law will be written additively throughout the paper. What we need is to form nP for large $n \in \mathbb{N}$ in a short amount of time. One way to do this is the *binary method* (cf. [12]). This method uses the operations of "doubling" and "adding P". If we write n in its binary representation, the number of doublings is fixed by $\lfloor \log_2 n \rfloor$ and each *one* in this representation corresponds to an addition. Thus the cost of the multiplication depends on the length of the binary representation of n and the number of ones in this representation. The goal of the methods presented in this paper is to decrease the cost by finding representations of integers containing few nonzero digits.

If addition and subtraction are equally costly in the underlying group, it makes sense to work with *signed binary representations*, i.e., binary representations with digits $\{0, \pm 1\}$. The advantage of these representations is their redundancy: in general, n has many different signed binary representations. Let n be written in a signed binary representation. Then the number of non-zero digits is called the *Hamming weight* of this representation. Since each non-zero digit causes a group addition (1 causes addition of P, -1 causes subtraction of P), one is interested in finding a representation of n having minimal Hamming weight. Such a minimal representation was exhibited by Reitwiesner [10]. Since it has no adjacent non-zero digits, this type of representation is often called *non-adjacent form* or NAF, for short. On average, only one third of the digits of a NAF is different from zero. Morain and Olivos [9] first observed that NAFs are useful for calculating nP for large n quickly.

Recently, Solinas [11] considered the problem of computing $mP+nQ$ at once, without computing each of the summands separately. Using unsigned binary representations of m and n this can

Date: August 5, 2003.

2000 *Mathematics Subject Classification.* Primary: 11A63; Secondary: 94A60, 68W40.

† This author is supported by the START-project Y96-MAT of the Austrian Science Fund.

‡ This author is supported by the grant S8307-MAT of the Austrian Science Fund.

* This author is supported by the grant NRF 2053748 of the South African National Research Foundation.

¶ This author is supported by the grant S8310-MAT of the Austrian Science Fund.

be done with the help of the operations "doubling" and "adding P, Q, or $P + Q$". We are again interested in diminishing the number of additions. Assume that the additions of the three quantities are equally costly. If we write the binary representations $m = \sum a_j 2^j$ and $n = \sum b_j 2^j$ in the form $\begin{smallmatrix} a_\ell \\ b_\ell \end{smallmatrix} \cdots \begin{smallmatrix} a_0 \\ b_0 \end{smallmatrix}$, the cost of the calculation of $mP + nQ$ depends on the number of non-zero columns in this joint representation. This number is called the *joint Hamming weight* of this (joint) representation. If addition and subtraction are equally costly, again by using signed representations of m and n, one can reduce the joint Hamming weight considerably (note that for signed representations we have to deal with the addition of $\pm P$, $\pm Q$, $\pm P \pm Q$ and $\pm P \mp Q$). One way to do this consists in writing m and n in their NAF. However, in the above mentioned paper, Solinas found an even "cheaper" way of representing m and n: the so called *Joint Sparse Form*. It turns out that his construction yields the minimal joint Hamming weight among all joint expansions of two numbers. In Grabner et al. [3] this concept was simplified and extended to the joint representation of $d \geq 2$ numbers and its properties are studied in detail. The representation used in [3] is therefore called *Simple Joint Sparse Form*, or SJSF for short.

The detailed definitions of joint NAFs and SJSF for d integers will be given in Section 2 and Section 3, respectively.

In all these algorithms we determined nP and $mP+nQ$ by doubling and adding some quantities. There is a modification of these algorithms by using so called *windows* or *window methods* (cf. for instance Gordon [2, Section 3] or Avanzi [1]). This is a rather easy concept. We will explain it for the case of the computation of $mP + nQ$ with m, n written in binary representation. First select a window size w. Then precompute all sums of the form $rP + sQ$ such that r and s have a binary representation of length at most w. Now we can compute $mP + nQ$ by multiplying by 2^w and adding one of the precomputed values. Of course, this makes the algorithms faster at the cost of precomputation tables. There are many ways to refine this concept and to consider adaptions which are suitable to special representations. An easy modification consists in jumping over zero vectors at the beginning of a window. If we use window methods where zero digits are forced after a bounded number of non-zero digits we may adapt the size of the window after each step in order to exploit these zeros. The latter modification is possible for instance in the case of SJSF. We will explain all this in more detail when we apply windows to our algorithms later.

In the present paper we are concerned with the joint representation of d-tuples of integers. In Section 2 we dwell upon joint NAFs with windows. In particular, we give a detailed analysis of the average cost of calculating linear combinations $n_1P_1 + \cdots + n_dP_d$ by examining the joint Hamming weight of joint NAFs. We give expressions of the average cost, its variance as well as its distribution. This extends and refines the work of Avanzi [1].

In Section 3 we perform a detailed runtime analysis of the SJSF of d integers using window methods. Contrary to the joint NAF the SJSF guarantees that after at most d non-zero columns (or digits) there occurs a zero column. It is natural to adapt the size of the windows dynamically in a way that we can expect zero columns at the beginning of each new window. In this way we can exploit the existing zeros in an optimal way. In this case it is a nontrivial problem to compute the size of the precomputation tables. We give an asymptotic formula for their size.

From the way we calculate the linear combinations $n_1P_1 + \cdots + n_dP_d$ we see that we proceed through the representations of $n_1, \ldots, n_d$ starting from their most significant digit down their least significant digit, or, in other words, *from left to right.* Unfortunately, as Avanzi [1] and Solinas [11] both regret, the known algorithms for the SJSF produce the representations from right to left. This has the disadvantage that we need to calculate the whole SJSF representation from right to left before we can start to apply it from left to right in order to compute our linear combinations. Especially if we have to deal with long representations this requires a large amount of memory.

Our last section, however, is devoted to a remedy to this unfortunate situation by providing a transducer (with 32 "essential" states) which constructs a minimal joint representation *from left to right* for $d = 2$. This is done by first writing each of the numbers m, n separately in a representation which gives us some freedom in changing their digits locally reading from the left. Because of its resemblance to the well-known greedy expansion we call this representation the *alternating greedy expansion.* Starting from this expansion we succeeded in constructing a minimal joint representation of m, n from left to right.

2. Joint non-adjacent form

The present section is devoted to the complexity analysis of the joint distribution of integers in non-adjacent form. Recall that a NAF is a signed binary representation of an integer x of the shape

$$x = \sum_{j=0}^{J} x_j 2^j \qquad \text{with} \quad x_j \in \{0, \pm 1\}$$

such that $x_j x_{j+1} = 0$ for all $j \in \{0, \dots, J-1\}$. For a given integer it is possible to compute its NAF with help of the easy Algorithm 1.

Algorithm 1 Calculation of the Non-Adjacent Form.

```
Input: $x$ integer
Output: $(x_j)_{0 \le j \le \ell}$ non-adjacent form of $x$.
  $j \leftarrow 0$
  while $x \neq 0$ do
    $x_j \leftarrow x \bmod 2$
    if $x_j = 1$ and $(x - x_j)/2 \equiv 1 \bmod 2$ then
      $x_j \leftarrow -x_j$
    end if
    $x \leftarrow (x - x_j)/2$
    $j \leftarrow j + 1$
  end while
```

Note that Algorithm 1 is the same as the algorithm for computing the simple joint sparse form for $d = 1$ (see Section 3). This algorithm can easily be interpreted as a three state transducer (cf. Figure 1). Using this transducer an easy calculation yields that the expected value of the

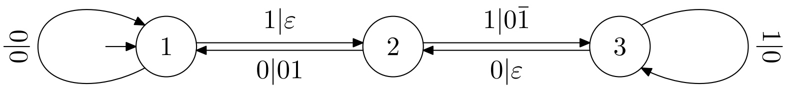

Figure 1. Transducer to compute the NAF from right to left.

Hamming weight of an expansion of length J is $J/3$.

Let $d \in \mathbb{N}$. In what follows we will investigate d-tuples of NAFs. Such a d-tuple will be called *joint NAF.* Joint NAFs can be regarded as finite sequences of d-dimensional vectors. We will write d-dimensional vectors in boldface. For the coordinates of a vector we will use the notational convention

$$\mathbf{x} = (x^{(1)}, \dots, x^{(d)}).$$

We will set up an easy probabilistic model. Define the space

$$\mathcal{N}_d := \left\{ (\dots, \mathbf{x}_1, \mathbf{x}_0) \in \{0, \pm 1\}^{d \times \mathbb{N}_0} \mid \forall j \in \mathbb{N}_0,\, k \in \{1, \dots, d\} : x_j^{(k)} x_{j+1}^{(k)} = 0 \right\}$$

whose elements will be called *infinite joint NAFs.* On $\mathcal{N}_d$ we define a probability measure by the image of the Haar measure on $\mathbb{Z}_2^d := \{0,1\}^{d \times \mathbb{N}_0}$ via the map $\mathbb{Z}_2^d \to \mathcal{N}_d$ given by Algorithm 1. The joint Hamming weight of a joint NAF $(\mathbf{x}_j)_{j \ge 0}$ is the number of $j \in \mathbb{N}_0$ with $\mathbf{x}_j \neq \mathbf{0}$. In order to derive results on the disribution of the Hamming weight of NAFs we need information on the number of nonzero entries in a vector $\mathbf{x}_j$. Thus we set

$$A(\mathbf{x}) := \left\{ k \in \{1, \dots, d\} \mid x^{(k)} \neq 0 \right\}. \tag{2.1}$$

Let $\mathbf{x}, \mathbf{y} \in \{0, \pm 1\}^d$ satisfying $x^{(k)}y^{(k)} = 0$ for all $k \in \{1, \dots, d\}$. We define the random variable $\mathbf{X}_j$ to be the j-th column of an infinite joint NAF. Then, keeping track of Algorithm 1 for each of the coordinates, we derive

$$\mathbb{P}(\mathbf{X}_{j+1} = \mathbf{y} \mid \mathbf{X}_j = \mathbf{x}) = 2^{-d-\#A(\mathbf{y})+\#A(\mathbf{x})} \tag{2.2}$$

and

$$\mathbb{P}(\mathbf{X}_0 = \mathbf{y}) = 2^{-d-\#A(\mathbf{y})}. \tag{2.3}$$

As mentioned above, we are only interested in the Hamming weight of joint NAFs. Thus it suffices to consider the random variables $\#A(\mathbf{X}_j)$ rather than $\mathbf{X}_j$ itself. Using (2.2) we easily derive

$$p_{k,\ell} := \mathbb{P}(\#A(\mathbf{X}_{j+1}) = \ell \mid \#A(\mathbf{X}_j) = k) = 2^{-d-k}\binom{d-k}{\ell}.$$

These quantities will be helpful in order to study the number of group additions required for multiple exponentiation algorithms which are used in cryptography (cf. Avanzi [1]). As mentioned in the introduction, such algorithms can be accelerated by using window methods (cf. for instance Gordon [2]). Suppose we want to compute the linear combination $x^{(1)}P_1 + \cdots + x^{(d)}P_d$ in an Abelian group G using joint NAFs with window length w. Then we need a table of precomputed values given by

$$\mathrm{PreComp}_{d,w} := \left\{ \sum_{j=0}^{w-1} 2^j \left(y_j^{(1)}P_1 + \cdots + y_j^{(d)}P_d \right) \mid \pm(\mathbf{y}_0, \dots, \mathbf{y}_{w-1}) \in \{0, \pm 1\}^{d \times w}/\{\pm 1\}, \mathbf{y}_0 \neq 0 \right\}.$$

It is clear that larger windows lead to less group additions at the cost of larger precomputation tables on the other hand. From Avanzi [1] we know that

$$\#\,\mathrm{PreComp}_{d,w} = \frac{I_w^d - I_{w-1}^d}{2} \qquad \text{with} \qquad I_w := \frac{2^{w+2} - (-1)^w}{3}.$$

This follows easily by noting that I_w is equal to the number of NAFs of length N, which can be computed by analyzing Algorithm 1.

We now want to examine Algorithm 2, which produces joint NAFs using windows. In particular, we want to derive distribution results for the random variable $W_{n,w}$ which counts the number of group additions in G when this algorithm is applied to $(\mathbf{X}_{n-1}, \dots, \mathbf{X}_0)$ (i.e., $W_{n,w}$ counts the number of windows that are "opened" by Algorithm 2, in other words, it counts how many group additions are required in order to compute a linear combination of group elements using the joint NAF). Since w is fixed we will write W_n instead of $W_{n,w}$.

To this matter we study the bivariate generating function

$$F(y,z) := \sum_{m=0}^{\infty} \sum_{n=0}^{\infty} \mathbb{P}(W_n = m) y^m z^n.$$

In order to get a closed expression for this function we note first that in view of Algorithm 2 each d-tuple of NAFs can be written using the regular expression

$$(0^* N B^{w-1})^* 0^* \{\varepsilon, N B^v\}. \tag{2.4}$$

Here $0 \leq v \leq w-2$, $B := \{0, \pm 1\}^d$, $N := B \setminus \{\mathbf{0}\}$ and ε is the empty word. In addition, each coordinate has to satisfy the NAF condition. Note that each occurrence of N in this regular expression causes a group addition in Algorithm 2. Thus, we have to label each occurrence of N with y. Labelling each digit with z will lead to the desired function. As usual, we encode the Markov chain defined by $p_{k,\ell}$ by matrices. Denote the $(d+1)$-dimensional identity matrix by I

Algorithm 2 Calculating linear combinations using joint NAF with windows.

```
Input: $P_1, \dots, P_d \in G$, $X \in \{0, \pm 1\}^{d \times (J+1)}$ joint NAF of $\mathbf{x} \in \mathbb{N}^d$, $w \in \mathbb{N}$, $\mathrm{PreComp}_{d,w}$
Output: $P = x_1 P_1 + \dots + x_d P_d$
  $P \leftarrow 0$
  $j \leftarrow J$
  while $j \geq 0$ do
    while $j \geq 0$ and $\mathbf{x}_j = \mathbf{0}$ do
      $P \leftarrow 2P$
      $j \leftarrow j - 1$
    end while
    if $j \geq w$ then
      $j \leftarrow j - w$
    else
      $w \leftarrow j$
      $j \leftarrow 0$
    end if
    for $k = 1, 2, \dots, d$ do
      $f^{(k)} \leftarrow \sum_{r=0}^{w-1} x_{j+r}^{(k)} 2^r$
    end for
    $s \leftarrow$ largest positive integer such that $2^s | f^{(k)}$ for all $k \in \{1, \dots, d\}$
    for $k = 1, 2, \dots, d$ do
      $f^{(k)} \leftarrow f^{(k)}/2^s$
    end for
    $P \leftarrow 2^{w-s} P$
    $P \leftarrow P + \sum_{k=1}^{d} f^{(k)} P_k$        {this can be looked up for free in $\mathrm{PreComp}_{d,w}$}
    $P \leftarrow 2^s P$
  end while
```

and set

$$\begin{aligned}
P &:= z(p_{k,\ell})_{0 \leq k,\ell \leq d}, \\
Z &:= z([\ell = 0] p_{k,\ell})_{0 \leq k,\ell \leq d}, \\
G &:= z([\ell > 0] p_{k,\ell})_{0 \leq k,\ell \leq d}, \\
L &:= yI,
\end{aligned}$$

where Iverson's notation, popularized in [5], has been used: $[P]$ is defined to be 1 if condition P is true, and 0 otherwise. By inspecting (2.4) we get

$$F(y,z) = (1, 0, \dots, 0) S(y,z)^{-1} T(y,z) (1, \dots, 1)^T \tag{2.5}$$

with

$$\begin{aligned}
S(y,z) &:= I - (I - Z)^{-1} LGP^{w-1}, \\
T(y,z) &:= (I - Z)^{-1} (I + LG(I - P^{w-1})(I - P)^{-1}).
\end{aligned}$$

We mention that S^{-1} represents the expression $(0^* NB^{w-1})^*$ in (2.4), while T represents the two possible tails after this expression. Note that the addition occurring in T encodes the two alternatives in (2.4). The vectors at the left and right hand side of the matrix expression for F can be explained as follows. Because of

$$\mathbb{P}(\mathbf{X}_1 = \mathbf{y}) = \mathbb{P}(\mathbf{X}_1 = \mathbf{y} \mid \mathbf{X}_j = \mathbf{0})$$

we always start with the digit vector $\mathbf{0}$. On the other hand, we can end up with an arbitrary digit vector.

One can easily imagine that the expression (2.5) becomes more and more complex for increasing dimensions d. Using `Mathematica`®, we computed F explicitly for $1 \leq d \leq 5$. To give an impression of the expressions obtained, we include the resulting generating function for $d = 1$:

$$F(y,z) = \frac{4y(-1+z)z^w - (-2)^w\bigl(-6+2yz^w+z(6+y(-3+z^w))\bigr)}{(-1+z)\bigl(-4y(-1+z)z^w+(-2)^w(-6+2yz^w+z(3+yz^w))\bigr)}.$$

Since these expressions become very large for $d \geq 2$, we refrain from writing them down here and refer to the file which is available at [4].

As usual, $\mathbb{E}(W_n)$ and $\mathbb{E}(W_n(W_n-1))$ are computed by taking the first and second derivative w.r.t. y, respectively, and setting $y = 1$. From this we can easily calculate the variance $\mathbb{V}(W_n)$. In view of (2.5) it is clear that for each choice of (d, w) we obtain a rational function F. The main term in the asymptotic expansion of $\mathbb{E}(W_n)$ and $\mathbb{V}(W_n)$ comes from the dominant double and triple pole of F, respectively. We state exactly those results which fit into one line, the others (main terms and constant terms for $\mathbb{E}(W_n)$ and $\mathbb{V}(W_n)$ and $1 \leq d \leq 5$) are available at [4]. For $d = 1$, we have

$$\begin{aligned}\mathbb{E}(W_n) &= \frac{3(-2)^w}{(-2)^w(4+3w)-4}n + \frac{3(-2)^w w(-8-(-2)^w+3(-2)^w w)}{2(-4+4(-2)^w+3(-2)^w w)^2} + O(\rho_w^n),\\ \mathbb{V}(W_n) &= \frac{12(5(-8)^w-4^w-4(-2)^w)}{((-2)^w(4+3w)-4)^3}n + O_w(1)\end{aligned}$$

for some $|\rho_w| < 1$. For $d = 2$, we get

$$\begin{aligned}\mathbb{E}(W_n) &= \frac{3(-2)^w(8+9(-2)^w)}{-16+16\cdot 4^w+24(-2)^w w+27\cdot 4^w w}n + O_w(1),\\ \mathbb{V}(W_n) &= \frac{48(-1+(-2)^w)(-2)^w(1+(-2)^w)(128+464(-2)^w+560\cdot 4^w+261(-8)^w)}{(-16+16\cdot 4^w+24(-2)^w w+27\cdot 4^w w)^3}n + O_w(1),\end{aligned}$$

and for $d = 3$,

$$\mathbb{E}(W_n) \sim \frac{9(-2)^w(16+36(-2)^w+37\cdot 4^w+21(-8)^w)}{-64-64(-2)^w+64(-8)^w+64\cdot 16^w+144(-2)^w w+324\cdot 4^w w+333(-8)^w w+189\cdot 16^w w}n.$$

We list these main terms for the pairs (d, w) with $1 \leq d \leq 5$ and $w \leq 3$ in Table 1.

Since the generating function $F(y, z)$ fits into the general scheme of H.-K. Hwang's "quasi-power theorem" (cf. [7]), the random variable W_n satisfies a central limit theorem

$$\lim_{n\to\infty} \mathbb{P}\left(W_n \leq \mathbb{E}(W_n) + x\sqrt{\mathbb{V}(W_n)}\right) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} e^{-\frac{t^2}{2}}\,dt.$$

Theorem 1. *Let $w \geq 1$, $d \geq 1$ and $W_{n,w}$ be the random variable counting the number of group additions when calculating $X_1P_1+\cdots+X_dP_d$ using Algorithm 2, where $X_1, \ldots, X_d$ are independent random variables uniformly distributed on $\{0, \ldots, 2^n-1\}$. Then*

$$\lim_{n\to\infty} \mathbb{P}\left(W_{n,w} \leq \mathbb{E}(W_{n,w}) + x\sqrt{\mathbb{V}(W_{n,w})}\right) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} e^{-\frac{t^2}{2}}\,dt,$$

where $\mathbb{E}(W_{n,w})$ and $\mathbb{V}(W_{n,w})$ are given in [4] for $d \in \{1,2,3,4,5\}$ and in Table 1 for $d \leq 5$ and $w \leq 3$.

We remark that the main terms of the expected values for $d \in \{1, 2, 3\}$ are given by Avanzi [1]. In contrast to his approximate approach, our generating function approach allows us to extract lower order information as well as higher moments and to prove a central limit theorem.

3. Simple Joint Sparse Form

In this section we will adapt the window method to exponentiation studied in [1] to the d-dimensional Simple Joint Sparse Form introduced in [3]. We shortly summarize the definition of this joint expansion of elements of $\mathbb{Z}^d$.

(d,w)	$\frac{1}{n}\mathbb{E}(W_{n,w})$		$\frac{1}{n}\mathbb{V}(W_{n,w})$	
$(1,1)$	$\frac{1}{3}$	≈ 0.333333	$\frac{2}{27}$	≈ 0.074074
$(1,2)$	$\frac{1}{3}$	≈ 0.333333	$\frac{2}{27}$	≈ 0.074074
$(1,3)$	$\frac{2}{9}$	≈ 0.222222	$\frac{2}{81}$	≈ 0.024691
$(2,1)$	$\frac{5}{9}$	≈ 0.555556	$\frac{4}{27}$	≈ 0.148148
$(2,2)$	$\frac{11}{27}$	≈ 0.407407	$\frac{80}{2187}$	≈ 0.036580
$(2,3)$	$\frac{32}{117}$	≈ 0.273504	$\frac{2464}{177957}$	≈ 0.013846
$(3,1)$	$\frac{19}{27}$	≈ 0.703704	$\frac{1064}{6561}$	≈ 0.162170
$(3,2)$	$\frac{131}{297}$	≈ 0.441077	$\frac{196210}{8732691}$	≈ 0.022468
$(3,3)$	$\frac{1082}{3645}$	≈ 0.296845	$\frac{1822366}{199290375}$	≈ 0.009144
$(4,1)$	$\frac{65}{81}$	≈ 0.802469	$\frac{41776}{295245}$	≈ 0.141496
$(4,2)$	$\frac{3469}{7533}$	≈ 0.460507	$\frac{242550560}{15832158831}$	≈ 0.015320
$(4,3)$	$\frac{22976}{74115}$	≈ 0.310005	$\frac{95487386176}{15078376202625}$	≈ 0.006333
$(5,1)$	$\frac{211}{243}$	≈ 0.868313	$\frac{644320}{5845851}$	≈ 0.110218
$(5,2)$	$\frac{32297}{68283}$	≈ 0.472987	$\frac{7065325354906}{648539908339455}$	≈ 0.010894
$(5,3)$	$\frac{11961398}{37601091}$	≈ 0.318113	$\frac{963646563298519282}{218773676422818911097}$	≈ 0.004405

TABLE 1. Asymptotic means and variances of $W_{n,w}/n$ for small d, w.

In [3] it is shown that for each d-tuple of integers $(x^{(1)}, \dots, x^{(d)})$ there is a unique joint expansion $(\mathbf{x}_J, \dots, \mathbf{x}_2, \mathbf{x}_1, \mathbf{x}_0)$, i.e.,

$$x^{(k)} = \sum_{j=0}^{J} x_j^{(k)} 2^j \qquad \text{with} \qquad x_j^{(k)} \in \{0, \pm 1\},$$

such that

$$A(\mathbf{x}_{j+1}) \supsetneq A(\mathbf{x}_j) \text{ or } A(\mathbf{x}_{j+1}) = \emptyset, \qquad 0 \le j < J \tag{3.1}$$

and $\mathbf{x}_J \neq \mathbf{0}$, where $A(\mathbf{x})$ has been defined in (2.1). This is called the *Simple Joint Sparse Form* of $x^{(1)}$, …, $x^{(d)}$.

3.1. Algorithms and probabilistic model.

Algorithms and probabilistic model. Algorithm 3 can be used for the computation of the Simple Joint Sparse Form of d integers. This algorithm was described in [3]; we will need this algorithm to derive the transition probabilities for the probabilistic model we will use.

From (3.1) it is clear that the Simple Joint Sparse Form can have at most d consecutive non-zero digit-vectors. When applying windows to the computation of $x^{(1)}P_1 + \cdots + x^{(d)}P_d$ in an Abelian group using the SJSF, it is natural to use these "guaranteed" 0 digit-vectors. Therefore we will consider Algorithm 4.

Let $\mathcal{J}_d$ be the space of all "infinite SJSFs", i.e.,

$$\mathcal{J}_d = \left\{ (\dots, \mathbf{x}_1, \mathbf{x}_0) \in \{0, \pm 1\}^{d \times \mathbb{N}_0} \mid \forall j \in \mathbb{N}_0 : A(\mathbf{x}_j) \subsetneq A(\mathbf{x}_{j+1}) \text{ or } A(\mathbf{x}_{j+1}) = \emptyset \right\}. \tag{3.2}$$

We now define a probability measure on $\mathcal{J}_d$ as the image of the Haar-measure on $\mathbb{Z}_2^d = \{0,1\}^{d \times \mathbb{N}_0}$ under the map $\mathbb{Z}_2^d \to \mathcal{J}_d$ given by Algorithm 3. This measure induces uniform distribution on all possible input vectors in $\{0, \dots, 2^N - 1\}^d$. Analyzing the congruence conditions used in Algorithm 3 yields (for $\mathbf{x}, \mathbf{y} \in \{0, \pm 1\}^d$ satisfying $A(\mathbf{x}) \subsetneq A(\mathbf{y})$ or $\mathbf{y} = \mathbf{0}$)

$$\mathbb{P}\left(\mathbf{X}_{j+1} = \mathbf{y} \mid \mathbf{X}_j = \mathbf{x}\right) = 2^{-(d + \#A(\mathbf{y}) - \#A(\mathbf{x}))} \tag{3.3}$$

Algorithm 3 d-dimensional Simple Joint Sparse Form.

```
Input: $x^{(1)}, \ldots, x^{(d)}$ integers
Output: $(x_j^{(k)})_{\substack{1\le k\le d\\ 0\le j\le \ell}}$ Simple Joint Sparse Form of $x^{(1)}, \ldots, x^{(d)}$
  $j \leftarrow 0$
  $A_0 \leftarrow \{k \mid x^{(k)} \text{ odd}\}$
  while $\exists k : x^{(k)} \neq 0$ do
    $x_j^{(k)} \leftarrow x^{(k)} \bmod 2$, $1 \le k \le d$
    $A_{j+1} \leftarrow \{k \mid (x^{(k)} - x_j^{(k)})/2 \equiv 1 \pmod 2\}$
    if $A_{j+1} \subseteq A_j$ then
      for all $k \in A_{j+1}$ do
        $x_j^{(k)} \leftarrow -x_j^{(k)}$
      end for
      $A_{j+1} \leftarrow \emptyset$
    else
      for all $k \in A_j \setminus A_{j+1}$ do
        $x_j^{(k)} \leftarrow -x_j^{(k)}$
      end for
      $A_{j+1} \leftarrow A_j \cup A_{j+1}$
    end if
    $x^{(k)} \leftarrow (x^{(k)} - x_j^{(k)})/2$, $1 \le k \le d$
    $j \leftarrow j + 1$
  end while
```

Algorithm 4 Calculating linear combinations using Simple Joint Sparse Forms with windows.

```
Input: $P_1, \ldots, P_d \in G$, $X \in \{0, \pm 1\}^{d\times(J+1)}$ SJSF of $(x_1, \ldots, x_d)$, $w \ge 1$ integer, $Q(Y) = \sum_{\ell=0}^{L} 2^\ell (y_\ell^{(1)} P_1 + \cdots + y_\ell^{(d)} P_d)$ for all SJSF $Y \in \mathrm{PreComp}_{d,w}$
Output: $P = x_1 P_1 + \cdots + x_d P_d$
  $P \leftarrow 0$
  $j \leftarrow J$
  while $j \ge 0$ do
    while $j \ge 0$ and $X_j = \mathbf{0}$ do
      $P \leftarrow 2P$
      $j \leftarrow j - 1$
    end while
    find $i$ such that $X_i = \mathbf{0}$ and such that there are exactly $w - 1$ $\mathbf{0}$-digit vectors amongst the digit vectors $X_j, X_{j-1}, \ldots, X_{i+1}$ or $i \leftarrow -1$
    find $k$ minimal with $i < k \le j$ and $X_k \neq \mathbf{0}$
    $P \leftarrow 2^{k-i}(2^{j-k} P \pm Q(\pm(X_j, X_{j-1}, \ldots, X_k)))$
    $j \leftarrow i$
  end while
```

and

$$\mathbb{P}(\mathbf{X}_0 = \mathbf{y}) = 2^{-(d + \#A(\mathbf{y}))}. \tag{3.4}$$

We are interested in the random variable $W_n = W_{n,w}(\mathbf{X})$, which counts the number of group additions when applying Algorithm 4 to $(\mathbf{X}_{n-1}, \ldots, \mathbf{X}_0)$. Since W_n depends only on

$$(\#A(\mathbf{X}_{n-1}), \ldots, \#A(\mathbf{X}_0)),$$

we compute the corresponding transition probabilities

$$p_{k,\ell} := \mathbb{P}\left(\#A(\mathbf{X}_{j+1}) = \ell \mid \#A(\mathbf{X}_j) = k\right) = \begin{cases} \binom{d-k}{\ell-k} 2^{-(d-k)} & \text{for } \ell > k, \\ 2^{-(d-k)} & \text{for } \ell = 0. \end{cases} \tag{3.5}$$

In order to study W_n we introduce the generating function

$$F(y,z) = \sum_{m=0}^{\infty} \sum_{n=0}^{\infty} \mathbb{P}\left(W_n = m\right) y^m z^n. \tag{3.6}$$

We denote $N = \{0, \pm 1\}^d \setminus \{\mathbf{0}\}$. A regular expression for a window containing $w-1$ "interior $\mathbf{0}$'s" is given by

$$NN^*(\mathbf{0}N^*)^{w-1}$$

(we remark here that the conditions (3.1) have to be satisfied). Since adjacent windows can be separated by an arbitrary number of $\mathbf{0}$s and windows at the end of the expansion can be incomplete, the generating function $F(y,z)$ can be calculated by

$$\begin{aligned} F(y,z) &= (1,0,\ldots,0)\Big(I - (I-V)^{-1} LU\big((I-U)^{-1}V\big)^w\Big)^{-1} (I-V)^{-1} \\ &\quad \times \Big(I + \big(LU + LU(I-U)^{-1}V + \cdots + LU((I-U)^{-1}V)^{w-1}\big)(I-U)^{-1}\Big)(1,\ldots,1)^T, \end{aligned} \tag{3.7}$$

where

$$\begin{aligned} U &= z([\ell > 0] p_{k,\ell})_{0 \le k,\ell \le d}, \\ V &= z([\ell = 0] p_{k,\ell})_{0 \le k,\ell \le d}, \\ L &= \operatorname{diag}(y, 1, \ldots, 1). \end{aligned}$$

We remark here that the "entry vector" $(1,0,\ldots,0)$ simulates a $\mathbf{0}$ in position -1 which corresponds to the fact that $\mathbb{P}(\mathbf{X}_0 = \mathbf{y})$ can be computed by setting $\mathbf{x} = \mathbf{0}$ in (3.3).

For $d \le 20$ we computed the function $F(y,z)$ using `Mathematica`®. Only the result for $d=2$ fits on one line:

$$F(y,z) = \frac{4 - (4-3y)z^3 + z\left(4 + y\left(3 - 2^{3-2w}\left(z(1+z)^2\right)^w\right)\right) - z^2\left(4 - y\left(6 - 4^{1-w}\left(z(1+z)^2\right)^w\right)\right)}{(1-z)\left(4 - z^3 + z\left(7 - 2^{3-2w}y\left(z(1+z)^2\right)^w\right) + z^2\left(2 - 4^{1-w}y\left(z(1+z)^2\right)^w\right)\right)}.$$

As usual, the generating functions of $\mathbb{E}(W_n)$ and $\mathbb{E}(W_n(W_n-1))$ are computed by differentiating $F(y,z)$ with respect to y and setting $y=1$. Since these functions are all rational, the main term in the asymptotic expansion of the moments of W_n comes from a double resp. triple pole at $z=1$; the other terms are exponentially smaller in magnitude. For $d = 1, \ldots, 20$ we computed the means and variances of W_n. Table 2 gives the asymptotic main terms of means and variances for $1 \le d \le 7$.

Since the generating function $F(y,z)$ fits into the general scheme of H.-K. Hwang's "quasi-power theorem" (cf. [7]), the random variable W_n satisfies a central limit theorem.

By the same means we compute expectation and variance of the number $W_{n,0}$ of additions using SJSFs without windows (see Table 3).

We summarize our results in the following theorem.

Theorem 2. *Let $w \ge 1$, $d \ge 1$ and $W_{n,w}$ be the random variable counting the number of group additions when calculating $X_1P_1 + \cdots + X_dP_d$ using Algorithm 4, where $X_1, \ldots, X_d$ are independent random variables uniformly distributed on $\{0, \ldots, 2^n - 1\}$. Then*

$$\lim_{n\to\infty} \mathbb{P}\left(W_{n,w} \le \mathbb{E}(W_{n,w}) + x\sqrt{\mathbb{V}(W_{n,w})}\right) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{t^2}{2}}\, dt,$$

where $\mathbb{E}(W_{n,w})$ and $\mathbb{V}(W_{n,w})$ are given in [4] *for $1 \le d \le 20$ and in Table 2 for $d \le 7$.*

d	$\frac{1}{n}\mathbb{E}(W_{n,w})$	$\frac{1}{n}\mathbb{V}(W_{n,w})$
1	$\dfrac{2}{3(w+1)}$	$\dfrac{2(w+7)}{27(w+1)^3}$
2	$\dfrac{3}{2(3w+1)}$	$\dfrac{9}{16(3w+1)^2}$
3	$\dfrac{112}{39(7w+1)}$	$\dfrac{784(1225w-137)}{59319(7w+1)^3}$
4	$\dfrac{960}{179(15w+1)}$	$\dfrac{8640(82175w-10751)}{5735339(15w+1)^3}$
5	$\dfrac{63488}{6327(31w+1)}$	$\dfrac{63488(3549810031w-337187183)}{253275687783(31w+1)^3}$
6	$\dfrac{4128768}{218357(63w+1)}$	$\dfrac{12386304(5319844735149w-322156222637)}{10411213601145293(63w+1)^3}$
7	$\dfrac{1065353216}{29681427(127w+1)}$	$\dfrac{1065353216(1110439852652223895w-40282349901979031)}{2614895455649204000 1483(127w+1)^3}$

Table 2. Asymptotic means and variances of $W_{n,w}$.

3.2. **Counting the precomputed values.** We now count the number of elements in the set $\mathrm{PreComp}_{d,w}$ of precomputed values. The following computations will show that $\#\,\mathrm{PreComp}_{d,w}$ increases exponentially in w and hyperexponentially in d. The set $\mathrm{PreComp}_{d,w}$ consists of all finite sequences of digit vectors satisfying (3.1), containing at most $w-1$ $\mathbf{0}$-digit vectors, and which start and end with a non-zero digit vector. Furthermore, we normalize the elements of $\mathrm{PreComp}_{d,w}$ by requiring that the first non-zero entry in the first column equals $+1$. Since any admissible sequence of digit vectors can be followed by an arbitrary number of $\mathbf{0}$-digit vectors, we have

$$\#\,\mathrm{PreComp}_{d,w} = \#\left(\left\{X \in (\{0,\pm1\}^d)^* \mid X \in NN^*(\mathbf{0}N^*)^{w-1} \text{ and } X \text{ a SJSF}\right\}/\{\pm1\}\right),$$

where $N = \{0,\pm1\}^d \setminus \{\mathbf{0}\}$. Thus we have for $w \geq 1$

$$\#\,\mathrm{PreComp}_{d,w} = \frac{1}{2}(1,0,\ldots,0)C(I-C)^{-1}(B(I-C)^{-1})^{w-1}(1,\ldots,1)^T,$$

where

$$B = ([\ell = 0])_{0\leq k,\ell\leq d}, \qquad C = ([\ell > k]\binom{d-k}{\ell-k}2^\ell)_{0\leq k,\ell\leq d}.$$

In order to study the behaviour for large d, we study the matrix $B(I-C)^{-1}$ in more detail. Since $\operatorname{rank} B = 1$ it is clear that all rows of $B(I-C)^{-1}$ are equal. It can be proved by induction that the k-th entry in this row equals

$$\binom{d}{k}2^k a_k,$$

where a_k satisfies the recursion

$$a_n = \sum_{k=0}^{n-1}\binom{n}{k}2^k a_k \quad n \geq 1, \quad a_0 = 1. \tag{3.8}$$

d	$\frac{1}{n}\mathbb{E}(W_{n,0})$	$\frac{1}{n}\mathbb{V}(W_{n,0})$
1	$\frac{1}{3}$	$\frac{2}{27}$
2	$\frac{1}{2}$	$\frac{1}{16}$
3	$\frac{23}{39}$	$\frac{2800}{59319}$
4	$\frac{115}{179}$	$\frac{210368}{5735339}$
5	$\frac{4279}{6327}$	$\frac{7565047808}{253275687783}$
6	$\frac{152821}{218357}$	$\frac{263523314106368}{10411213601145293}$
7	$\frac{21292819}{29681427}$	$\frac{577533922219434967040}{2614895455649204000 1483}$

Table 3. Asymptotic means and variances of the number of additions when using SJSF without windows.

Since $\lambda_d = \sum_{k=0}^{d} dk2^k a_k = (2^d+1)a_d$ is the only non-zero eigenvalue of $B(I-C)^{-1}$, we get $\#\,\mathrm{PreComp}_{d,w} = \frac{1}{2}(\lambda_d - 1)\lambda_d^{w-1}$.

In order to study the asymptotic behaviour of a_n we substitute $a_n = n!2^{\binom{n}{2}}b_n$. This is motivated by the fact that the dominating summand in (3.8) occurs for $k = n-1$; this gives $a_n \approx n2^{n-1}a_{n-1}$. We get the recursion

$$b_n = \sum_{k=0}^{n-1} \frac{1}{(n-k)!} 2^{\binom{k+1}{2}-\binom{n}{2}} b_k \tag{3.9}$$

for b_n. Since the term for $k = n-1$ on the right hand side equals b_{n-1}, the sequence b_n is monotonically increasing. It remains to show that b_n is bounded. For this purpose we estimate

$$b_n \le b_{n-1}\left(1 + \sum_{k=0}^{n-2} \frac{1}{(n-k)!} 2^{\binom{k+1}{2}-\binom{n}{2}}\right) \le b_{n-1}\left(1 + \sum_{k=2}^{n} \frac{1}{k!} 2^{-\frac{1}{2}n(k-1)}\right),$$

where we have used $\binom{n}{2} - \binom{k+1}{2} \ge \frac{1}{2}n(n-k-1)$. Using $\frac{e^x-1}{x} \le e^x$ for $x > 0$ and extending the finite sums on the right hand side to infinite sums, we obtain

$$b_n \le b_{n-1} \exp\left(2^{-\frac{1}{2}n}\right) \le b_0 \exp\left(\sum_{k=1}^{n} 2^{-\frac{k}{2}}\right).$$

Thus b_n is bounded and we can form the generating function

$$f(z) = \sum_{n=0}^{\infty} b_n z^n. \tag{3.10}$$

Inserting the recursion (3.9) into (3.10) yields

$$f(z) = 1 + \sum_{n=1}^{\infty} \sum_{k=0}^{n-1} \frac{1}{(n-k)!} 2^{\binom{k+1}{2} - \binom{n}{2}} b_k z^n = 1 + \sum_{\ell=1}^{\infty} \frac{1}{\ell!} 2^{-\binom{\ell}{2}} z^\ell \sum_{k=0}^{\infty} b_k \left(2^{-(\ell-1)} z\right)^k. \tag{3.11}$$

The inner sum in (3.11) is just $f(2^{-(\ell-1)}z)$. Furthermore, the summand for $\ell = 1$ equals $zf(z)$. This yields

$$f(z) = \frac{1}{1-z} \left(1 + \sum_{\ell=1}^{\infty} \frac{1}{(\ell+1)!2^{\binom{\ell+1}{2}}} z^{\ell+1} f(2^{-\ell} z)\right), \tag{3.12}$$

from which we conclude that f has a meromorphic continuation to the whole complex plane with simple poles at $z = 2^\ell$, $\ell \in \mathbb{N}$. The residue of $f(z)$ at $z = 1$ equals

$$\begin{aligned} c = 1 + \sum_{\ell=1}^{\infty} \frac{1}{(\ell+1)!2^{\binom{\ell+1}{2}}} f(2^{-\ell}) \\ = 1.57298\,62035\,88985\,42167\,40408\,30458\,77385\,46604\,92965\ldots. \end{aligned}$$

Putting everything together we conclude that

$$a_n \sim cn!2^{\binom{n}{2}}. \tag{3.13}$$

Summing up, we have

Theorem 3. *Let $d, w \geq 1$. Then the number of precomputed values $\#\operatorname{PreComp}_{d,w}$ needed in Algorithm 4 is given by*

$$\#\operatorname{PreComp}_{d,w} = \frac{1}{2}(\lambda_d - 1)\lambda_d^{w-1} \quad \textit{with} \quad \lambda_d \sim cd!2^{\binom{d+1}{2}}. \tag{3.14}$$

In order to compare the number of additions when computing linear combinations using SJSF with and without windows, we introduce the notation

$$\#\operatorname{PreComp}_{d,0} = \#\left(\left(\{0, \pm 1\}^d \setminus \{\mathbf{0}\}\right) / \{\pm 1\}\right) = \frac{3^d - 1}{2}$$

for the number of precomputations needed for SJSF without windows. Table 4 shows the number of precomputed values in the various situations.

d	$\#\operatorname{PreComp}_{d,w} - d$	$\#\operatorname{PreComp}_{d,0} - d$
1	$3^{w-1} - 1$	0
2	$12 \cdot 25^{w-1} - 2$	2
3	$301 \cdot 603^{w-1} - 3$	10
4	$19320 \cdot 38641^{w-1} - 4$	36

TABLE 4. Number of precomputed values.

Table 5 shows the minimal values of n, such that

$$\operatorname{PreComp}_{d,w} + \mathbb{E}(W_{n,w}) \leq \operatorname{PreComp}_{d,w-1} + \mathbb{E}(W_{n,w-1}).$$

d	$w=1$	$w=2$	$w=3$
1	1	19	110
2	65	1 793	112 001
3	1 249	1 081 666	1 793 670 987
4	62 748	4 602 740 129	511 331 633 697 389

Table 5. Threshold numbers n depending on the window size w and dimension d.

4. Calculating a minimal expansion from left to right

It is a major disadvantage of Joint Sparse Form representations of pairs of integers that they can only be computed reading the binary expansion from right to left (cf. [3, 11]). However, computing linear combinations using these representations requires the digits from left to right. Therefore, the full representation has to be precomputed and stored.

However, as in the one dimensional case (cf. [6]), it is possible to compute some minimal joint expansion from left to right using a transducer automaton.

The idea is as follows:

We want to work with redundant expansions of numbers that—when proceeding from left to right—always leave *alternatives.* For instance, if the number is 29, and we started with 1???? (where "?" stands for an arbitrary digit $0, \pm1$ not yet computed), then the next two positions are already forced, 111??, and only then one has choices: 11101 resp. $1111\bar{1}$; if we would have considered 31 instead, we would have no choice at all. That is undesirable, since we want to perform *local changes* in order to create as many $\genfrac{}{}{0pt}{}{0}{0}$'s as possible, and so we must always have alternatives. It is therefore wise to start the representation of 29 as 1?????. A good strategy is to alternatively over- resp. undershoot the number, which means that—ignoring intermediate zeros—the digit $\bar{1}$ follows 1 and vice versa. With that condition, the process is essentially unique, but we can still write 4 as 100 or as $1\bar{1}00$, etc. We found it easier to work with the second variant. The representation of 29 would then be $100\bar{1}01$. We call this representation *alternating greedy* expansion.

Definition 1. $\boldsymbol{\varepsilon} \in \{-1, 0, 1\}^{\mathbb{N}_0}$ is called an *alternating greedy expansion of* $n \in \mathbb{Z}$, if the following conditions are satisfied:

(1) Only a finite number of ε_j is nonzero,
(2) $n = \sum_{j \geq 0} \varepsilon_j 2^j$,
(3) If $\varepsilon_j = \varepsilon_\ell \neq 0$ for some $j < \ell$, then there is a k with $j < k < \ell$ such that $\varepsilon_j = -\varepsilon_k = \varepsilon_\ell$.
(4) For $\underline{j} := \min\{j : \varepsilon_j \neq 0\}$ and $\overline{j} := \max\{j : \varepsilon_j \neq 0\}$, we have $\operatorname{sign}(n) = \varepsilon_{\overline{j}} = -\varepsilon_{\underline{j}}$.

Proposition 1. *For any integer n, there is a unique alternating greedy expansion. It can be computed by Algorithm 5.*

The proof is easy. □

This "alternating greedy" expansion of single integers can be computed from the standard binary expansion by a transducer automaton as shown in Figure 2.

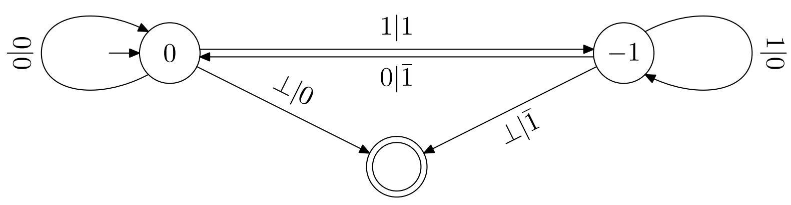

Figure 2. Transducer Automaton for computing an alternating greedy expansion from left to right. The symbol $\perp$ denotes the end of the sequence.

Now we think about a pair of integers (x, y), both given in the alternating greedy expansion. When we parse this two rowed representation from left to right, we claim the following: if we

Algorithm 5 Alternating Greedy Expansion.

```
Input: $n$ positive integer.
Output: Alternating greedy expansion $\boldsymbol{\varepsilon}$ of $n$.
  $m \leftarrow n$
  $\boldsymbol{\varepsilon} \leftarrow \mathbf{0}$
  while $m \neq 0$ do
    $k \leftarrow \lfloor \log_2 m \rfloor$
    if $m = -2^k$ then
      $\varepsilon_k = -1$
      Return($\boldsymbol{\varepsilon}$)
    else
      $\varepsilon_{k+1} \leftarrow \operatorname{sign}(m)$
      $m \leftarrow m - \varepsilon_{k+1} 2^{k+1}$
    end if
  end while
```

see $\begin{smallmatrix} a & c & e & \cdots \\ b & d & f & \cdots \end{smallmatrix}$, either $a = b = 0$, and we found a $\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}$, or if not then $\begin{smallmatrix} a & c \\ b & d \end{smallmatrix}$ can be rewritten such that $\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}$ is produced, or if this is not possible, $\begin{smallmatrix} a & c & e \\ b & d & f \end{smallmatrix}$ can be rewritten such that $\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}$ is produced. So, after at most three digits, a $\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}$ has been produced. In other words, a representation of length J has the property that at least $\lfloor J/3 \rfloor$ digits are equal to $\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}$. Recall that the representation of Solinas [11] (SJF) resp. the representation in [3] (SJSF) have this property. The Table 6 contains all the necessary information. (The obvious variants when either interchanging the top resp. bottom rows or exchanging $1 \leftrightarrow \bar{1}$ are not explicitly stated.) Note that in a few instances one would have some choices. E. g., if we see $\begin{smallmatrix} 1 & 0 & \bar{1} \\ 0 & 1 & \bar{1} \end{smallmatrix}$, we decided to replace it by $\begin{smallmatrix} 1 & 0 & \bar{1} \\ 0 & 0 & 1 \end{smallmatrix}$, but we could have chosen $\begin{smallmatrix} 0 & 1 & 1 \\ 0 & 1 & \bar{1} \end{smallmatrix}$ or $\begin{smallmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \end{smallmatrix}$ with the same effect.

Read	Write
$\begin{matrix} 0 \\ 0 \end{matrix}$	$\begin{matrix} 0 \\ 0 \end{matrix}$
$\begin{matrix} 1 & 0 \\ 1 & 0 \end{matrix}$	$\begin{matrix} 1 & 0 \\ 1 & 0 \end{matrix}$
$\begin{matrix} 1 & \bar{1} \\ 1 & \bar{1} \end{matrix}$	$\begin{matrix} 0 & 1 \\ 0 & 1 \end{matrix}$
$\begin{matrix} 1 & \bar{1} & 0 \\ 1 & 0 & 0 \end{matrix}$	$\begin{matrix} 1 & \bar{1} & 0 \\ 1 & 0 & 0 \end{matrix}$
$\begin{matrix} 1 & \bar{1} & 0 \\ 1 & 0 & \bar{1} \end{matrix}$	$\begin{matrix} 0 & 1 & 0 \\ 0 & 1 & 1 \end{matrix}$
$\begin{matrix} 1 & \bar{1} & 1 \\ 1 & 0 & 0 \end{matrix}$	$\begin{matrix} 1 & 0 & \bar{1} \\ 1 & 0 & 0 \end{matrix}$
$\begin{matrix} 1 & \bar{1} & 1 \\ 1 & 0 & \bar{1} \end{matrix}$	$\begin{matrix} 0 & 1 & 1 \\ 0 & 1 & 1 \end{matrix}$
$\begin{matrix} 1 & \bar{1} \\ 0 & x \end{matrix}$	$\begin{matrix} 0 & 1 \\ 0 & x \end{matrix}$
$\begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{matrix}$	$\begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{matrix}$
$\begin{matrix} 1 & 0 & 0 \\ 0 & 1 & \bar{1} \end{matrix}$	$\begin{matrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{matrix}$
$\begin{matrix} 1 & 0 & \bar{1} \\ 0 & 1 & 0 \end{matrix}$	$\begin{matrix} 0 & 1 & 1 \\ 0 & 1 & 0 \end{matrix}$
$\begin{matrix} 1 & 0 & \bar{1} \\ 0 & 1 & \bar{1} \end{matrix}$	$\begin{matrix} 1 & 0 & \bar{1} \\ 0 & 0 & 1 \end{matrix}$

TABLE 6. Rules for modifying a pair of alternating greedy expansions to a minimal joint expansion.

Clearly, this can be realized by a transducer automaton, too.

Combining the two transducer automata, we get a single transducer automaton to calculate a low weight expansion from the binary expansions of two integers. The resulting transducer automaton is shown in Table 7 and in Figure 3 (note that the final state has not been drawn in Figure 3).

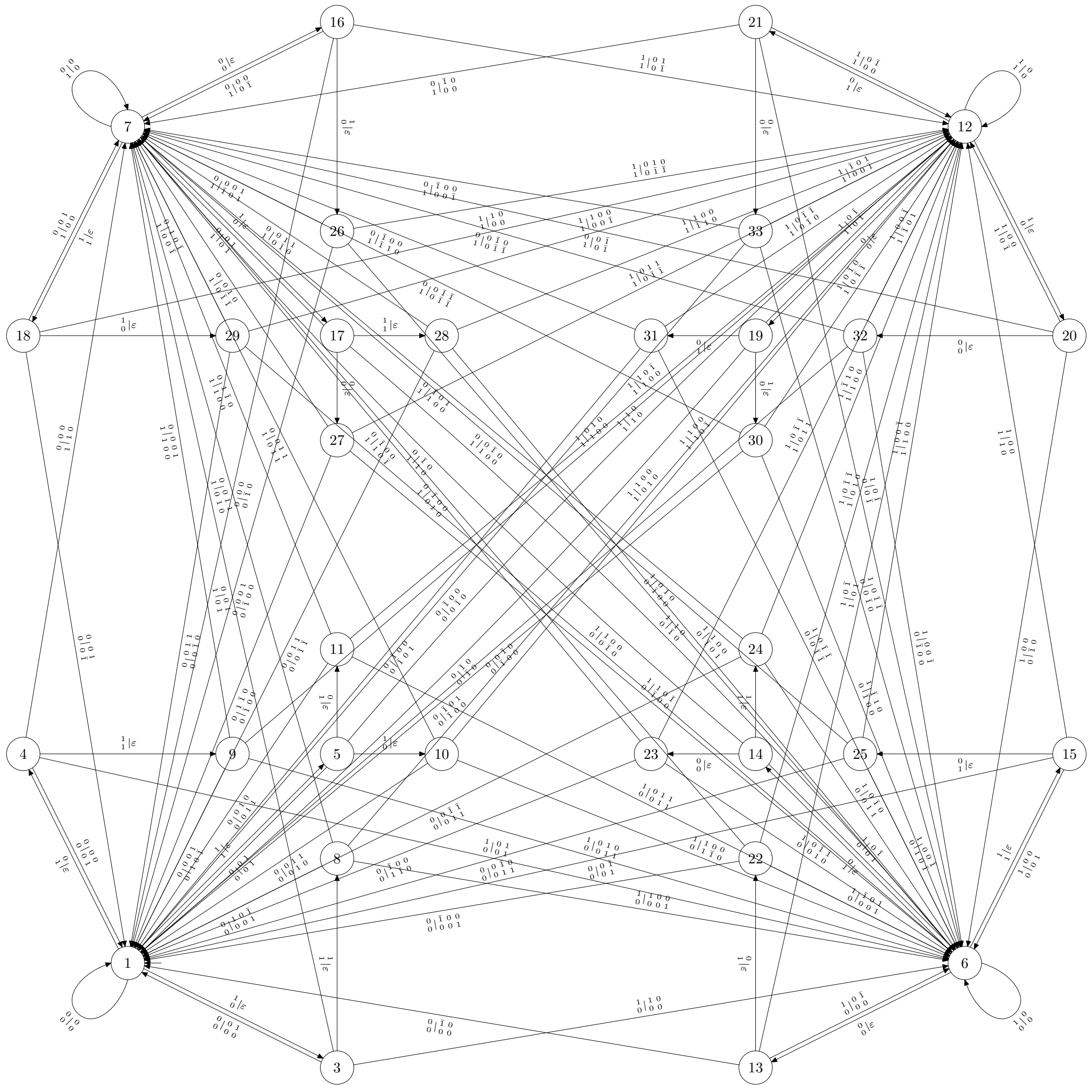

FIGURE 3. Transducer automaton for calculating a minimial joint expansion from the binary expansion.

We now prove that this transducer indeed calculates a minimal joint expansion. To this aim, we denote by $h(x,y)$ and $\widetilde{h}(x,y)$ the joint Hamming weights of the SJSF and the output of the transducer for x and y, respectively. We set

$$\begin{aligned}
w_{kn} &:= \#\{(x,y) \mid 0 \le x,y < 2^n, h(x,y)=k\},\\
\widetilde{w}_{kn} &:= \#\{(x,y) \mid 0 \le x,y < 2^n, \widetilde{h}(x,y)=k\},\\
F(Y,Z) &:= \sum_{k,n\ge 0} w_{kn} Y^k Z^n,\\
\widetilde{F}(Y,Z) &:= \sum_{k,n\ge 0} \widetilde{w}_{kn} Y^k Z^n.
\end{aligned}$$

		In: $\begin{smallmatrix}0\\0\end{smallmatrix}$		In: $\begin{smallmatrix}1\\0\end{smallmatrix}$		In: $\begin{smallmatrix}0\\1\end{smallmatrix}$		In: $\begin{smallmatrix}1\\1\end{smallmatrix}$		In: $\perp$	
State	Nr	Out:	To:	Out:	To:	Out:	To:	Out:	To:	Out:	To:
$\begin{smallmatrix}0\\0\end{smallmatrix}$	1	$\begin{smallmatrix}0\\0\end{smallmatrix}$	1	ε	3	ε	4	ε	5	$\begin{smallmatrix}0\\0\end{smallmatrix}$	2
$\perp$	2	ε	2	ε	2	ε	2	ε	2	ε	2
$\begin{smallmatrix}1&\bar1\\0&0\end{smallmatrix}$	3	$\begin{smallmatrix}0&1\\0&0\end{smallmatrix}$	1	$\begin{smallmatrix}1&0\\0&0\end{smallmatrix}$	6	$\begin{smallmatrix}0&1\\0&1\end{smallmatrix}$	7	ε	8	$\begin{smallmatrix}0&1\\0&0\end{smallmatrix}$	2
$\begin{smallmatrix}0&0\\1&\bar1\end{smallmatrix}$	4	$\begin{smallmatrix}0&0\\0&1\end{smallmatrix}$	1	$\begin{smallmatrix}0&1\\0&1\end{smallmatrix}$	6	$\begin{smallmatrix}0&0\\1&0\end{smallmatrix}$	7	ε	9	$\begin{smallmatrix}0&0\\0&1\end{smallmatrix}$	2
$\begin{smallmatrix}1&\bar1\\1&\bar1\end{smallmatrix}$	5	$\begin{smallmatrix}0&1\\0&1\end{smallmatrix}$	1	ε	10	ε	11	$\begin{smallmatrix}1&0\\1&0\end{smallmatrix}$	12	$\begin{smallmatrix}0&1\\0&1\end{smallmatrix}$	2
$\begin{smallmatrix}\bar1\\0\end{smallmatrix}$	6	ε	13	$\begin{smallmatrix}0\\0\end{smallmatrix}$	6	ε	14	ε	15	$\begin{smallmatrix}\bar1\\0\end{smallmatrix}$	2
$\begin{smallmatrix}0\\\bar1\end{smallmatrix}$	7	ε	16	ε	17	$\begin{smallmatrix}0\\0\end{smallmatrix}$	7	ε	18	$\begin{smallmatrix}0\\\bar1\end{smallmatrix}$	2
$\begin{smallmatrix}1&0&\bar1\\0&1&\bar1\end{smallmatrix}$	8	$\begin{smallmatrix}1&0&\bar1\\0&0&1\end{smallmatrix}$	1	$\begin{smallmatrix}1&0&0\\0&0&1\end{smallmatrix}$	6	$\begin{smallmatrix}0&1&1\\0&1&0\end{smallmatrix}$	7	$\begin{smallmatrix}1&0&0\\0&1&0\end{smallmatrix}$	12	$\begin{smallmatrix}1&0&\bar1\\0&0&1\end{smallmatrix}$	2
$\begin{smallmatrix}0&1&\bar1\\1&0&\bar1\end{smallmatrix}$	9	$\begin{smallmatrix}0&0&1\\1&0&\bar1\end{smallmatrix}$	1	$\begin{smallmatrix}0&1&0\\0&1&1\end{smallmatrix}$	6	$\begin{smallmatrix}0&0&1\\1&0&0\end{smallmatrix}$	7	$\begin{smallmatrix}0&1&0\\1&0&0\end{smallmatrix}$	12	$\begin{smallmatrix}0&0&1\\1&0&\bar1\end{smallmatrix}$	2
$\begin{smallmatrix}1&0&\bar1\\1&\bar1&0\end{smallmatrix}$	10	$\begin{smallmatrix}0&1&1\\0&1&0\end{smallmatrix}$	1	$\begin{smallmatrix}1&0&0\\1&\bar1&0\end{smallmatrix}$	6	$\begin{smallmatrix}0&1&1\\0&1&1\end{smallmatrix}$	7	$\begin{smallmatrix}1&0&0\\1&0&\bar1\end{smallmatrix}$	12	$\begin{smallmatrix}0&1&1\\0&1&0\end{smallmatrix}$	2
$\begin{smallmatrix}1&\bar1&0\\1&0&\bar1\end{smallmatrix}$	11	$\begin{smallmatrix}0&1&0\\0&1&1\end{smallmatrix}$	1	$\begin{smallmatrix}0&1&1\\0&1&1\end{smallmatrix}$	6	$\begin{smallmatrix}1&\bar1&0\\1&0&0\end{smallmatrix}$	7	$\begin{smallmatrix}1&0&\bar1\\1&0&0\end{smallmatrix}$	12	$\begin{smallmatrix}0&1&0\\0&1&1\end{smallmatrix}$	2
$\begin{smallmatrix}\bar1\\\bar1\end{smallmatrix}$	12	ε	19	ε	20	ε	21	$\begin{smallmatrix}0\\0\end{smallmatrix}$	12	$\begin{smallmatrix}\bar1\\\bar1\end{smallmatrix}$	2
$\begin{smallmatrix}\bar1&0\\0&0\end{smallmatrix}$	13	$\begin{smallmatrix}\bar1&0\\0&0\end{smallmatrix}$	1	$\begin{smallmatrix}0&\bar1\\0&0\end{smallmatrix}$	6	ε	22	$\begin{smallmatrix}0&\bar1\\0&1\end{smallmatrix}$	12	$\begin{smallmatrix}\bar1&0\\0&0\end{smallmatrix}$	2
$\begin{smallmatrix}\bar1&0\\1&\bar1\end{smallmatrix}$	14	ε	23	$\begin{smallmatrix}0&\bar1\\0&1\end{smallmatrix}$	6	$\begin{smallmatrix}\bar1&0\\1&0\end{smallmatrix}$	7	ε	24	$\begin{smallmatrix}\bar1&0\\1&\bar1\end{smallmatrix}$	2
$\begin{smallmatrix}0&\bar1\\1&\bar1\end{smallmatrix}$	15	$\begin{smallmatrix}0&\bar1\\0&1\end{smallmatrix}$	1	$\begin{smallmatrix}0&0\\0&1\end{smallmatrix}$	6	ε	25	$\begin{smallmatrix}0&0\\1&0\end{smallmatrix}$	12	$\begin{smallmatrix}0&\bar1\\0&1\end{smallmatrix}$	2
$\begin{smallmatrix}0&0\\\bar1&0\end{smallmatrix}$	16	$\begin{smallmatrix}0&0\\\bar1&0\end{smallmatrix}$	1	ε	26	$\begin{smallmatrix}0&0\\0&\bar1\end{smallmatrix}$	7	$\begin{smallmatrix}0&1\\0&\bar1\end{smallmatrix}$	12	$\begin{smallmatrix}0&0\\\bar1&0\end{smallmatrix}$	2
$\begin{smallmatrix}1&\bar1\\\bar1&0\end{smallmatrix}$	17	ε	27	$\begin{smallmatrix}1&0\\\bar1&0\end{smallmatrix}$	6	$\begin{smallmatrix}0&1\\0&\bar1\end{smallmatrix}$	7	ε	28	$\begin{smallmatrix}1&\bar1\\\bar1&0\end{smallmatrix}$	2
$\begin{smallmatrix}1&\bar1\\0&\bar1\end{smallmatrix}$	18	$\begin{smallmatrix}0&1\\0&\bar1\end{smallmatrix}$	1	ε	29	$\begin{smallmatrix}0&1\\0&0\end{smallmatrix}$	7	$\begin{smallmatrix}1&0\\0&0\end{smallmatrix}$	12	$\begin{smallmatrix}0&1\\0&\bar1\end{smallmatrix}$	2
$\begin{smallmatrix}\bar1&0\\\bar1&0\end{smallmatrix}$	19	$\begin{smallmatrix}\bar1&0\\\bar1&0\end{smallmatrix}$	1	ε	30	ε	31	$\begin{smallmatrix}0&\bar1\\0&\bar1\end{smallmatrix}$	12	$\begin{smallmatrix}\bar1&0\\\bar1&0\end{smallmatrix}$	2
$\begin{smallmatrix}0&\bar1\\\bar1&0\end{smallmatrix}$	20	ε	32	$\begin{smallmatrix}0&0\\\bar1&0\end{smallmatrix}$	6	$\begin{smallmatrix}0&\bar1\\0&\bar1\end{smallmatrix}$	7	$\begin{smallmatrix}0&0\\0&\bar1\end{smallmatrix}$	12	$\begin{smallmatrix}0&\bar1\\\bar1&0\end{smallmatrix}$	2
$\begin{smallmatrix}\bar1&0\\0&\bar1\end{smallmatrix}$	21	ε	33	$\begin{smallmatrix}0&\bar1\\0&\bar1\end{smallmatrix}$	6	$\begin{smallmatrix}\bar1&0\\0&0\end{smallmatrix}$	7	$\begin{smallmatrix}0&\bar1\\0&0\end{smallmatrix}$	12	$\begin{smallmatrix}\bar1&0\\0&\bar1\end{smallmatrix}$	2
$\begin{smallmatrix}\bar1&0&0\\0&1&\bar1\end{smallmatrix}$	22	$\begin{smallmatrix}\bar1&0&0\\0&0&1\end{smallmatrix}$	1	$\begin{smallmatrix}\bar1&0&1\\0&0&1\end{smallmatrix}$	6	$\begin{smallmatrix}\bar1&0&0\\0&1&0\end{smallmatrix}$	7	$\begin{smallmatrix}0&\bar1&\bar1\\0&1&0\end{smallmatrix}$	12	$\begin{smallmatrix}\bar1&0&0\\0&0&1\end{smallmatrix}$	2
$\begin{smallmatrix}\bar1&0&0\\1&\bar1&0\end{smallmatrix}$	23	$\begin{smallmatrix}\bar1&0&0\\1&\bar1&0\end{smallmatrix}$	1	$\begin{smallmatrix}0&\bar1&\bar1\\0&1&0\end{smallmatrix}$	6	$\begin{smallmatrix}\bar1&0&0\\1&0&\bar1\end{smallmatrix}$	7	$\begin{smallmatrix}0&\bar1&\bar1\\0&1&1\end{smallmatrix}$	12	$\begin{smallmatrix}\bar1&0&0\\1&\bar1&0\end{smallmatrix}$	2
$\begin{smallmatrix}\bar1&1&\bar1\\1&0&\bar1\end{smallmatrix}$	24	$\begin{smallmatrix}0&\bar1&\bar1\\0&1&1\end{smallmatrix}$	1	$\begin{smallmatrix}0&\bar1&0\\0&1&1\end{smallmatrix}$	6	$\begin{smallmatrix}\bar1&0&1\\1&0&0\end{smallmatrix}$	7	$\begin{smallmatrix}\bar1&1&0\\1&0&0\end{smallmatrix}$	12	$\begin{smallmatrix}0&\bar1&\bar1\\0&1&1\end{smallmatrix}$	2
$\begin{smallmatrix}0&\bar1&0\\1&0&\bar1\end{smallmatrix}$	25	$\begin{smallmatrix}0&\bar1&0\\0&1&1\end{smallmatrix}$	1	$\begin{smallmatrix}0&0&\bar1\\1&0&\bar1\end{smallmatrix}$	6	$\begin{smallmatrix}0&\bar1&0\\1&0&0\end{smallmatrix}$	7	$\begin{smallmatrix}0&0&\bar1\\1&0&0\end{smallmatrix}$	12	$\begin{smallmatrix}0&\bar1&0\\0&1&1\end{smallmatrix}$	2
$\begin{smallmatrix}0&1&\bar1\\\bar1&0&0\end{smallmatrix}$	26	$\begin{smallmatrix}0&0&1\\\bar1&0&0\end{smallmatrix}$	1	$\begin{smallmatrix}0&1&0\\\bar1&0&0\end{smallmatrix}$	6	$\begin{smallmatrix}0&0&1\\\bar1&0&1\end{smallmatrix}$	7	$\begin{smallmatrix}0&1&0\\0&\bar1&\bar1\end{smallmatrix}$	12	$\begin{smallmatrix}0&0&1\\\bar1&0&0\end{smallmatrix}$	2
$\begin{smallmatrix}1&\bar1&0\\\bar1&0&0\end{smallmatrix}$	27	$\begin{smallmatrix}1&\bar1&0\\\bar1&0&0\end{smallmatrix}$	1	$\begin{smallmatrix}1&0&\bar1\\\bar1&0&0\end{smallmatrix}$	6	$\begin{smallmatrix}0&1&0\\0&\bar1&\bar1\end{smallmatrix}$	7	$\begin{smallmatrix}0&1&1\\0&\bar1&\bar1\end{smallmatrix}$	12	$\begin{smallmatrix}1&\bar1&0\\\bar1&0&0\end{smallmatrix}$	2
$\begin{smallmatrix}1&0&\bar1\\\bar1&1&\bar1\end{smallmatrix}$	28	$\begin{smallmatrix}0&1&1\\0&\bar1&\bar1\end{smallmatrix}$	1	$\begin{smallmatrix}1&0&0\\\bar1&0&1\end{smallmatrix}$	6	$\begin{smallmatrix}0&1&1\\0&\bar1&0\end{smallmatrix}$	7	$\begin{smallmatrix}1&0&0\\\bar1&1&0\end{smallmatrix}$	12	$\begin{smallmatrix}0&1&1\\0&\bar1&\bar1\end{smallmatrix}$	2
$\begin{smallmatrix}1&0&\bar1\\0&\bar1&0\end{smallmatrix}$	29	$\begin{smallmatrix}0&1&1\\0&\bar1&0\end{smallmatrix}$	1	$\begin{smallmatrix}1&0&0\\0&\bar1&0\end{smallmatrix}$	6	$\begin{smallmatrix}1&0&\bar1\\0&0&\bar1\end{smallmatrix}$	7	$\begin{smallmatrix}1&0&0\\0&0&\bar1\end{smallmatrix}$	12	$\begin{smallmatrix}0&1&1\\0&\bar1&0\end{smallmatrix}$	2
$\begin{smallmatrix}\bar1&1&\bar1\\\bar1&0&0\end{smallmatrix}$	30	$\begin{smallmatrix}\bar1&0&1\\\bar1&0&0\end{smallmatrix}$	1	$\begin{smallmatrix}\bar1&1&0\\\bar1&0&0\end{smallmatrix}$	6	$\begin{smallmatrix}0&\bar1&\bar1\\0&\bar1&\bar1\end{smallmatrix}$	7	$\begin{smallmatrix}0&\bar1&0\\0&\bar1&\bar1\end{smallmatrix}$	12	$\begin{smallmatrix}\bar1&0&1\\\bar1&0&0\end{smallmatrix}$	2
$\begin{smallmatrix}\bar1&0&0\\\bar1&1&\bar1\end{smallmatrix}$	31	$\begin{smallmatrix}\bar1&0&0\\\bar1&0&1\end{smallmatrix}$	1	$\begin{smallmatrix}0&\bar1&\bar1\\0&\bar1&\bar1\end{smallmatrix}$	6	$\begin{smallmatrix}\bar1&0&0\\\bar1&1&0\end{smallmatrix}$	7	$\begin{smallmatrix}0&\bar1&\bar1\\0&\bar1&0\end{smallmatrix}$	12	$\begin{smallmatrix}\bar1&0&0\\\bar1&0&1\end{smallmatrix}$	2
$\begin{smallmatrix}0&\bar1&0\\\bar1&0&0\end{smallmatrix}$	32	$\begin{smallmatrix}0&\bar1&0\\\bar1&0&0\end{smallmatrix}$	1	$\begin{smallmatrix}0&0&\bar1\\\bar1&0&0\end{smallmatrix}$	6	$\begin{smallmatrix}0&\bar1&0\\0&\bar1&\bar1\end{smallmatrix}$	7	$\begin{smallmatrix}0&0&\bar1\\\bar1&0&1\end{smallmatrix}$	12	$\begin{smallmatrix}0&\bar1&0\\\bar1&0&0\end{smallmatrix}$	2
$\begin{smallmatrix}\bar1&0&0\\0&\bar1&0\end{smallmatrix}$	33	$\begin{smallmatrix}\bar1&0&0\\0&\bar1&0\end{smallmatrix}$	1	$\begin{smallmatrix}0&\bar1&\bar1\\0&\bar1&0\end{smallmatrix}$	6	$\begin{smallmatrix}\bar1&0&0\\0&0&\bar1\end{smallmatrix}$	7	$\begin{smallmatrix}\bar1&0&1\\0&0&\bar1\end{smallmatrix}$	12	$\begin{smallmatrix}\bar1&0&0\\0&\bar1&0\end{smallmatrix}$	2

TABLE 7. Transducer automaton for calculating a minimial joint expansion from the binary expansion from left to right. The symbol $\perp$ denotes the end of the sequence.

Let $\widetilde{m}^{(\delta\varepsilon)} := Y^h$, where h is the joint Hamming weight of the transition $j = \binom{\delta}{\varepsilon}.i$ in the transducer. If there is no such transition, we set $\widetilde{m}^{(\delta\varepsilon)} := 0$. We set $\widetilde{M}^{(\delta,\varepsilon)} := (\widetilde{m}_{ij})_{1\le i,j\le 32}$ and $\widetilde{M} := \sum_{0\le\delta,\varepsilon\le 1} M^{(\delta,\varepsilon)}$. Then it is easily seen that

$$\widetilde{F}(Y,Z) = \big(1,0,\dots,0\big)\cdot(I - Z\widetilde{M})^{-1}\cdot(\widetilde{M}^{(00)})^2\cdot\big(1,0,\dots,0\big)^T.$$

The factors $M^{(00)}$ ensure that we return to state 1 writing all accumulated digits. An explicit calculation yields

$$\widetilde{F}(Y,Z) = \frac{-1+(2-3Y)Z+(-1+13Y-9Y^2)Z^2+Y(-10+37Y-18Y^2)Z^3-2Y^2(16-23Y+8Y^2)Z^4+8Y^3(-4+3Y)Z^5}{(-1+Z)(-1+Z+2YZ^2)(-1+Z+8YZ^2+16Y^2Z^3)}.$$

A similar calculation using the right-to-left transducer described in [3] yields

$$\widetilde{F}(Y,Z) = F(Y,Z).$$

This implies $\widetilde{w}_{kn} = w_{kn}$ for all nonnegative k and n. Since $\widetilde{h}(x,y) \ge h(x,y)$ for all x, y, this proves that $\widetilde{h}(x,y) = h(x,y)$, as required.

So we proved the following theorem.

Theorem 4. *Let x, $y\in\mathbb{Z}$ with binary expansions $x = \sum_{j=0}^{J} x_j 2^j$ and $y = \sum_{j=0}^{J} y_j 2^j$. Then the output $(\varepsilon_{J+1}\dots\varepsilon_0)$ of the transducer in Table 7 when reading $\binom{x_J}{y_J}\dots\binom{x_0}{y_0}\perp)$ is a joint expansion of x and y of minimal joint Hamming weight.*

References

[1] R. Avanzi, *On multi-exponentiation in cryptography*, 2003, Preprint, available at `http://citeseer.nj.nec.com/545130.html`.

[2] D. M. Gordon, *A survey of fast exponentiation methods*, J. Algorithms **27** (1998), no. 1, 129–146.

[3] P. J. Grabner, C. Heuberger, and H. Prodinger, *Distribution results for low-weight binary representations for pairs of integers*, Preprint, available at `http://www.opt.math.tu-graz.ac.at/~cheub/publications/Joint_Sparse.pdf`.

[4] P. J. Grabner, C. Heuberger, H. Prodinger, and J. Thuswaldner, *Analysis of linear combination algorithms in cryptography — online resources*, `http://www.opt.math.tu-graz.ac.at/~cheub/publications/ECLinComb/`.

[5] R. L. Graham, D. E. Knuth, and O. Patashnik, *Concrete mathematics. A foundation for computer science*, second ed., Addison-Wesley, 1994.

[6] C. Heuberger, *Minimal expansions in redundant number systems: Fibonacci bases and greedy algorithms*, Preprint.

[7] H.-K. Hwang, *On convergence rates in the central limit theorems for combinatorial structures*, European J. Combin. **19** (1998), 329–343.

[8] N. Koblitz, A. Menezes, and S. Vanstone, *The state of elliptic curve cryptography*, Des. Codes Cryptogr. **19** (2000), no. 2-3, 173–193, Towards a quarter-century of public key cryptography.

[9] F. Morain and J. Olivos, *Speeding up the computations on an elliptic curve using addition-subtraction chains*, Inform Theory Appl. **24** (1990), 531–543.

[10] G. W. Reitwiesner, *Binary arithmetic*, Advances in computers, Vol. 1, Academic Press, New York, 1960, pp. 231–308.

[11] J. A. Solinas, *Low-weight binary representations for pairs of integers*, Tech. Report CORR 2001-41, University of Waterloo, 2001, available at `http://www.cacr.math.uwaterloo.ca/techreports/2001/corr2001-41.ps`.

[12] J. von zur Gathen and J. Gerhard, *Modern computer algebra*, Cambridge University Press, New York, 1999.

(P. Grabner) Institut für Mathematik A, Technische Universität Graz, Steyrergasse 30, 8010 Graz, Austria

E-mail address: `peter.grabner@tugraz.at`

(C. Heuberger) Institut für Mathematik B, Technische Universität Graz, Steyrergasse 30, 8010 Graz, Austria

E-mail address: `clemens.heuberger@tugraz.at`

(H. Prodinger) The John Knopfmacher Centre for Applicable Analysis and Number Theory, School of Mathematics, University of the Witwatersrand, P. O. Wits, 2050 Johannesburg, South Africa

E-mail address: `helmut@maths.wits.ac.za`

(J. Thuswaldner) Institut für Mathematik und Angewandte Geometrie, Montanuniversität Leoben, Franz-Josef-Strasse 18, 8700 Leoben, Austria

E-mail address: `Joerg.Thuswaldner@unileoben.ac.at`