On the BMS Algorithm

Shojiro Sakata
The University of Electro-Communications
Department of Information and Communication Engineering
Chofu-shi, Tokyo 182-8585, JAPAN

Abstract

I will present a sketch of the N-dimensional (N-D) Berlekamp-Massey algorithm (alias Berlekamp-Massey-Sakata or BMS algorithm). I mention: (1) How is it related to Gröbner basis? (2) What problem can it solve? (3) How does it work? First I will discuss a problem closely related to ours, and introduce some concepts about N-D linear recurrences and modules of N-D arrays as their general solutions. The former problem and ours are just the inverse to each other, which can be solved by Buchberger algorithm and BMS algorithm, respectively. Furthermore, I discuss some properties of BMS algorithm and its outputs, including its computational complexity.

1 Introduction

In this lecture, I will present a sketch of the multidimensional Berlekamp-Massey algorithm (alias Berlekamp-Massey-Sakata algorithm or BMS algorithm) [1][2]. It is a generalization of Berlekamp-Massey algorithm [3][4]. I mention:

(1) How is it related to Gröbner basis theory?

(2) What problem can it solve?

(3) How does it work?

In another session I will mention its application to decoding of algebraic error-correcting codes [5]. In most part of these lectures we restrict ourselves to treating finite fields although the contents remain valid in any field provided that we have exact computations.

Before we introduce our main theme, i.e. our problem and its solution by BMS algorithm, we discuss a problem closely related to ours as well as some concepts which are important in this paper. It also will turn to be a history of Gröbner basis in the world of Coding Theory.

Now we start to consider a simple sequence or (one-dimensional) array and a linear recurrence satisfied by it. The following is the the linear recurrence satisfied by the famous Fibonacci sequence:

$$s_{j+2} - s_{j+1} - s_j = 0,\ j \geq 0$$

Over the real number field, when we start with the initial values $s_0 = 1, s_1 = 1$, we have the one-dimensional (1-D) array $(s_j) = (1, 1, 2, 3, 5, 8, \cdots)$ (Over a finite field we have another array, of course). Well, we generalize such 1-D arrays and 1-D linear recurrences to multi-dimensional arrays and multidimensional linear recurrences. For example, we consider the following system of two-dimensional (2-D) linear recurrences:

$$\begin{cases} u_{i+2,j} + u_{i,j} = 0 \\ u_{1+1,j+1} + u_{i,j} = 0, \quad (i,j) \in \mathbf{Z}_0^2, \\ u_{i,j+2} + u_{i,j} = 0 \end{cases} \tag{1}$$

where $\mathbf{Z}_0$ is the set of nonzero integers. In general such a condition as above is called a system of constant-coefficient *linear recurrence*s or *(partial) finite difference equation*s. Given a system of N-dimensional (N-D) linear recurrences over a finite field $\mathbf{F}_q$, we want to find all N-D arrays satisfying them. It is just a digital version of finding the general solutions of a system of (homogeneous) constant-coefficient linear partial differential equations. We want to obtain not only a special solution but also the general solutions (the whole set of solutions). We treat multiple recurrences satisfied by N-D arrays. To discuss our problem in general we need some notation as follows.

For a fixed integer N we consider N-D arrays $u = (u_{\underline{i}})_{\underline{i} \in \mathbf{Z}_0^N}$ with elements $u_{\underline{i}} \in \mathbf{F}_q$ arranged on the whole N-D integral lattice $\mathbf{Z}_0^N := \{\underline{i} = (i_1, i_2, \cdots, i_N) \mid i_j \in \mathbf{Z}_0,\ 1 \leq j \leq N\}$. Let A be the set of all N-D arrays over $\mathbf{F}_q$ defined on $\mathbf{Z}_0^N$, and introduce basic operations

upon arrays $u \in A$. Naturally, we have the sum of two arrays $u = (u_{\underline{i}}), v = (v_{\underline{i}}) \in A$ as $u + v = (u_{\underline{i}} + v_{\underline{i}}) \in A$, and the scalar product of u by an element c of the coefficient field $\mathbf{F}_q$ as $cu = (cu_{\underline{i}}) \in A$. Furthermore, we consider polynomials $f = \sum_{\underline{i} \in \mathrm{Supp}(f)} f_{\underline{i}} \underline{x}^{\underline{i}} \in R$, where $R := \mathbf{F}_q[\underline{x}] := \mathbf{F}_q[x_1, \cdots, x_N]$ is the N-variate polynomial ring over $\mathbf{F}_q$, and we denote $\underline{x}^{\underline{i}} := x_1^{i_1} \cdots, x_N^{i_N}$. We call the set of *degrees* $\underline{i}$ ($\in \mathbf{Z}_0^N$) of its nonzero terms $f_{\underline{i}} \underline{x}^{\underline{i}}$ (having the nonzero coefficient $f_{\underline{i}} \in \mathbf{F}_q \setminus \{0\}$) by the name of the *support* of f and denote it as $\mathrm{Supp}(f)$ ($\subset \mathbf{Z}_0^N$). (Remark: In case of $N \geq 2$, the exponent $\underline{i}$ of a monomial $\underline{x}^{\underline{i}}$ is an integer vector, which we call *degree* for simplicity.) A polynomial $f \in R$ is operated on an array $u \in A$ so that the following array v is obtained:

$$\begin{aligned} v &= f \circ u := (v_{\underline{i}}) \in A \\ v_{\underline{i}} &:= \sum_{\underline{k} \in \mathrm{Supp}(f)} f_{\underline{k}} u_{\underline{k}+\underline{i}},\ \underline{i} \in \mathbf{Z}_0^N \end{aligned}$$

This *polynomial operation* '$f\circ$' is just a transformation of an array u into another array v. In particular, the operation by the monomial $f = x_j$, $1 \leq j \leq N$ is a unit shift along the x_j-axis (to the negative direction), where $v = x_j \circ u = (v_{\underline{i}})$, $\underline{i} \in \mathbf{Z}_0^N$ has the components $v_{i_1,\cdots,i_j,\cdots,i_N} = u_{i_1,\cdots,i_j+1,\cdots,i_N}$ (the components of u which are put out of the domain $\mathbf{Z}_0^N$ are pruned away). For example, in case of $N = 1$, for $x := x_1$ and $u = (u_i)$, the unit-shifted array $v = x \circ u = (v_i)$ has the components $v_i = u_{i+1}$, $i \in \mathbf{Z}_0$, and the double-shifted array $w = x^2 \circ u = (w_i)$ has $w_i = u_{i+2}$, $i \in \mathbf{Z}_0$, etc. (Trivially, by multiplying a polynomial $g = \sum_{0 \leq i \leq d} g_i x^i$ with x, one gets the polynomial $\bar{g} = xg = \sum_{0 \leq i \leq d} g_i x^{i+1} = \sum_{1 \leq i \leq d+1} g_{i-1} x^i$, where the array of coefficients of its terms is obtained by shifting to the positive direction: $(g_i) \to (g_{i-1})$ in contrast with the above shift (to the negative direction) by operation x.) Consequently, the module A is an R-module, i.e. a module with the ring R of operators. By using this notation, we can write any linear recurrences with the *characteristic polynomials* $F = \{f^{(1)}, \cdots, f^{(L)}\}$ ($\subset R$) as follows,

$$f^{(k)} \circ u = 0,\ 1 \leq k \leq L, \tag{2}$$

where 0 is the all-zero array. From now on, we do not distinguish between linear recurrences and the corresponding characteristic polynomials, identifying them. That is, for simplicity, provided that the formula (2) holds, we often say that the array u satisfies the polynomial $f^{(k)}$, and that 'the polynomial $f^{(k)}$ is *valid* for the array u,' etc. For a given $F \subset R$, it is easy to see that the set $A(F)$ of solutions u of (2) is an R-submodule of the R-module A, since $f \circ (g \circ u) = (fg) \circ u$ for $f, g \in R$. For example, for a univariate polynomial $f = x^2 - x - 1$, $A(f)$ is the set of 1-D arrays (Fibonacci sequences) $u = (u_i)$ which are obtained by setting any initial values u_0, u_1 and then uniquely by determining the other values u_i, $i \geq 2$ iteratively with the linear recurrence $f \circ u = 0$. In general, for any polynomial set $F \subset \mathbf{F}_q[\underline{x}]$ and the ideal $I(F) := \langle F \rangle_R$ generated by F, $A(F) = A(I(F)) := \{u \in A \mid f \circ u = 0,\ f \in I(F)\}$.

As is shown, in case of $N = 1$, we can easily obtain R-modules $A(f)$ and $A(F)$ of arrays. In particular, for $F = \{f^{(1)}, \cdots, f^{(L)}\}$ ($\subset \mathbf{F}_q[x]$), $A(F) = A(g)$, where $g = \gcd(F)$ (gcd=greatest common divisor). However, it is not so easy to obtain $A(F)$ in case of $N \geq 2$ as in case of $N = 1$. In case of $N \geq 2$, it is difficult to specify even the positions of initial values. The Gröbner basis theory provides us with the notion of *orders* over $\mathbf{Z}_0^N$ for discussing this issue. For $N \geq 2$, in addition to the natural *partial order* $\leq_P$ over $\mathbf{Z}_0^N$:

$$\underline{i} \leq_P \underline{j} \quad \Leftrightarrow \quad i_k \leq j_k, 1 \leq k \leq N$$

we have the *total order* defined with a fixed *weight* vector $\underline{w} = (w_k)$ ($\neq 0$) $\in \mathbf{Z}_0^N$:

$$\begin{aligned} \underline{i} \leq_T \underline{j} \Leftrightarrow \sum_{1 \leq k \leq N} w_k i_k < \sum_{1 \leq k \leq N} w_k j_k \quad &\vee \\ (\sum_{1 \leq k \leq N} w_k i_k = \sum_{1 \leq k \leq N} w_k j_k \quad &\wedge \quad \underline{i} \leq_L \underline{j}), \end{aligned}$$

where $\leq_L$ is any appropriate lexicographic order. This order $\leq_T$ satisfies a kind of stable condition as below and sometimes is called *monomial order* as is well-known:

$$\underline{i} \leq_T \underline{j} \quad \Rightarrow \quad \underline{i} + \underline{k} \leq_T \underline{j} + \underline{k}$$

According to this total order, we have a regular arrangement of terms of a polynomial $f = \sum_{\underline{i} \in \mathrm{Supp}(f)} f_{\underline{i}} \underline{x}^{\underline{i}} \in \mathbf{F}_q[\underline{x}]$ so that we can define the *degree*, which is usually called multidegree, the *head coefficient* and the *head term* of f as follows ($\max_T S$ is the maximum element of S ($\subset \mathbf{Z}_0^N$) w.r.t. $\leq_T$):

$$\deg(f) := \max_T \mathrm{Supp}(f) \quad (\in \mathbf{Z}_0^N)$$

$$\begin{aligned}\mathrm{hc}(f) &:= f_{\deg(f)} \quad (\in \mathbf{F}_q \setminus \{0\})\\ \mathrm{ht}(f) &:= f_{\deg(f)}\underline{x}^{\deg(f)}\end{aligned}$$

We consider the problem of initial value positions via the following example in case of $N = 2$. Now we assume for a set of polynomials $F = \{f^{(1)}, \cdots, f^{(L)}\}$ $(\subset \mathbf{F}_q[x_1, x_2])$ that the degrees $\deg(f^{(l)}) = \underline{d}^{(l)} = (d_1^{(l)}, d_2^{(l)}) \in \mathbf{Z}_0^2$, $1 \le l \le L$ of its elements satisfy

$$\begin{aligned} d_1^{(1)} &> d_1^{(2)} > \cdots > d_1^{(L-1)} > d_1^{(L)} = 0,\\ d_2^{(1)} = 0 &< d_2^{(2)} < \cdots < d_2^{(L-1)} < d_2^{(L)}\end{aligned}$$

Then, $\mathbf{Z}_0^2$ can be split into two parts:

$$\begin{aligned}\Sigma_F &:= \{\underline{i} \in \mathbf{Z}_0^2 \mid \underline{i} \ge_P \underline{d}^{(l)}, 1 \le^{\exists} l \le L\}\\ \Delta_F &:= \mathbf{Z}_0^2 \setminus \Sigma_F\end{aligned}$$

These subsets have the follwoing properties and are called *stable* sets (Sometimes the former and latter sets are called *upper* and *lower set*s, respectively),

$$\begin{aligned}\underline{i} \in \Sigma_F, \underline{j} \in \mathbf{Z}_0^2, \underline{i} \le_P \underline{j} &\Rightarrow \underline{j} \in \Sigma_F\\ \underline{i} \in \Delta_F, \underline{j} \in \mathbf{Z}_0^2, \underline{i} \ge_P \underline{j} &\Rightarrow \underline{j} \in \Delta_F\end{aligned}$$

among of which the set Δ_F called *delta-set* seemingly can be used as the initial value positions. That is, after having specified any values $u_{\underline{j}} \in \mathbf{F}_q$, $\underline{j} \in \Delta_F$ as the initial values, we proceed to find each of the remaining values $u_{\underline{j}}$, $\underline{j} \in \Sigma_F$ iteratively by using the following algorithm of generating an array up to an prescribed position. In the following we denote the next (w.r.t. the total order $\le_T$) point of any point $\underline{i} \in \mathbf{Z}_0^2$ as $\underline{i} \oplus 1$, and define $\overline{\mathrm{Supp}}(f) := \mathrm{Supp}(f) \setminus \{\deg(f)\}$ $(\subset \mathbf{Z}_0^2)$, $\Sigma^{\underline{r}} := \{\underline{j} \in \mathbf{Z}_0^2 \mid \underline{j} <_T \underline{r}\}$.

Algorithm 1 *Generating an array*

Input*:*

a polynomial set F with the delta-set Δ_F;

the initial values $u_{\underline{j}}$ $(\in \mathbf{F}_q)$, $\underline{j} \in \Delta_F$;

the terminal point $\underline{r}$;

Output*:*

an array $u_{\underline{j}}$, $\underline{j} \in \Sigma^{\underline{r}}$;

Procedure

Step 1 (initialization): $\underline{j} := \min_T \Sigma_F$ *(the minimum element of Σ_F w.r.t. $\le_T$);*

Step 2 (computation): if $\underline{j} \in \Sigma_F$ then

begin

let l be any l, $1 \le l \le L$ s.t. $\underline{d}^{(l)} \le_P \underline{j}$;

$$u_{\underline{j}} := \frac{1}{\mathrm{hc}(f^{(l)})}(- \sum_{\underline{k} \in \overline{\mathrm{Supp}}(f^{(l)})} f_{\underline{k}}^{(l)} u_{\underline{k}+\underline{j}-\underline{d}^{(l)}});$$

end;

Step 3 (termination): $\underline{j} := \underline{j} \oplus 1$*;*

if $\underline{j} <_T \underline{r}$ then go to Step 2 else stop.

Although it seems that we could find an array u before the terminal point $\underline{r}$ by using this algorithm, it will turn out that we do not always succeed to get a proper array having the specified initial values and satisfying all of the given linear recurrences. In Step 2, the value $u_{\underline{j}}$ is determined by using the polynomial $f^{(l)}$ so that one of the desired condition

$$f^{(l)}[u]_{\underline{j}} := \sum_{\underline{k} \in \mathrm{Supp}(f^{(l)})} f_{\underline{k}}^{(l)} u_{\underline{k}+\underline{j}-\underline{d}^{(l)}} = 0$$

is satisfied, but some conditions corresponding to other polynomials $f^{(l')}$, $l' \ne l$ with $\underline{d}^{(l')} \le_P \underline{j}$ might not always be satisfied. Consider the previous example (1) over $\mathbf{F}_2$. The linear recurrences are specified by $F = \{f^{(1)} := x_1^2 + 1, f^{(2)} := x_1x_2 + 1, f^{(3)} := x_2^2 + 1\}$, and the delta-set is $\Delta_F = \{(0,0), (1,0), (0,1)\}$. Starting with the initial values shown in Table 1, we proceed to find the other values of u iteratively w.r.t. the total order $\le_T$ with the weight $\underline{w} = (1,1)$ (associated with the lexicographic order $x_1 <_L x_2$).

Table 1: initial values

$j_1 \setminus j_2$	0	1	2	3
0	1	0		
1	1			
2				
3				
4				

Then, we have the intermediate result shown in Table 2.

Table 2: partial array

$j_1 \setminus j_2$	0	1	2	3
0	1	0	1	
1	1	1		
2	1	0^*		
3	1			
4				

The value $u_{2,1} = 0$ with signature * is found by using $f^{(1)}$, but it does not satisfy the linear recurrence of $f^{(2)}$. Thus, there exists no array u which has the initial values and is a solution of the linear recurrences (1). In other words, the above delta-set is not appropriate as a set of positions for initial values. On taking into consideration the properties of Gröbner basis, it might be hit upon that the polynomial set F is not a Gröbner basis and that the corresponding delta-set is too large to be a set of positions for initial values so that the algorithm for generating arrays fails to find proper arrays. In fact, it is easy to see that the reduced Gröbner basis (w.r.t. the total order $\leq_T$ with $\underline{w} = (1,1)$) of the ideal $I = \langle F \rangle_R$ is $\{x_1^2 + 1, x_2 + x_1\}$, and that the proper set of positions for initial values is $\{(0,0),(1,0)\}$. As we have seen, the problem of finding the proper set of positions for initial values, given a set of linear recurrences or its characteristic polynomials, is closely related with that of finding a Gröbner basis of the ideal $I(F)$. Furthermore, we can find a polynomial $f^{(\underline{j})} := \underline{x}^{\underline{j}} - \sum_{\underline{i} \in \Delta_F} f^{(\underline{j})} \underline{x}^{\underline{i}} \in I(F)$ by which we can get the value $u_{\underline{j}}$ for any $\underline{j} \in \Sigma_F$ directly from any given values $u_{\underline{j}}$, $\underline{j} \in \Delta_F$ so that the so-called S-polynomial at any corner point outside Δ_F can be obtained. By repeating reductions of such S-polynomials modulo F and consequent modifications of F and $\Delta(F)$, we finally get a Gröbner basis. Of course, it is just the Buchberger algorithm. Let us illustrate it with the previous example. We get $x_2 f^{(1)} = x_1^2 x_2 + x_2 \in I(F)$ from $f^{(1)} = x_1^2 + 1$, and $x_1 f^{(2)} = x_1^2 x_2 + x_1$ from $f^{(2)} = x_1 x_2 + 1$. Then, we obtain the S-polynomial $f^{(3)} := x_2 f^{(1)} - x_1 f^{(2)} = x_2 + x_1 \in I(F)$ at the corner point $(2,1)$, which is not reducible further modulo F. Finally, $\{f^{(1)}, f^{(3)}\}$ turns out to be the reduced Gröbner basis. As a summary, we have that the problem of finding a module of linear recurrences and that of finding a Gröbner basis of its characteristic polynomials are equivalent with each other, and thus they can be solved by the same algorithm. In the world of Coding Theory, it is a historical fact that the concept of an equivalent of Gröbner basis and an equivalent of Buchberger algorithm were introduced in the process of solving a certain problem of constructing a kind of multidimensional codes.

2 BMS algorithm

Our main problem is just the inverse of the problem of finding the general solutions of a given system of linear recurrences which we have discussed in the previous section. Now, for integers L and N, we are given a pair of sets $U := \{u^{(l)}\}$ and $V := \{v^{(l)}\}$, $1 \leq l \leq L$ of infinite (periodic) N-D arrays, and consider the following linear recurrences:

$$f \circ u^{(l)} = v^{(l)},\ 1 \leq l \leq L$$

We might be required to find a polynomial f having a minimal degree $\deg(f)$. In this paper we concentrate upon the homogeneous problem with the right-hand side arrays $v^{(l)} = 0$, $1 \leq l \leq L$, leaving the non-homogeneous problem. It is easy to see that the following set of polynomials is an ideal of $R = \mathbf{F}_q[\underline{x}]$, which we call the *characteristic ideal* of the given set U of arrays $u^{(l)}$, $1 \leq l \leq L$:

$$I(U) := \{f \in R \mid f \circ u^{(l)} = 0,\ 1 \leq l \leq L\}.$$

For simplicity, we treat only the case of a single array, i.e. $U = \{u\}$, and we will give a method of finding a Gröbner basis of the characteristic ideal $I(u) := \{f \in R \mid f \circ u = 0\}$. We want to find a set of polynomials $f = \sum_{\underline{k} \in \mathrm{Supp}(f)} f_{\underline{k}} \underline{x}^{\underline{k}}$ with a minimal degree $\underline{d} = \deg(f)$ which satisfy $f \circ u = 0$ or

$$f[u]_{\underline{j}} := \sum_{\underline{k} \in \mathrm{Supp}(f)} f_{\underline{k}} u_{\underline{k}+\underline{j}-\underline{d}} = 0,\ \underline{j} \geq_P \underline{d} \tag{3}$$

We try to find a set of polynomials with a minimal degree satisfying a (partial) condition specified by a finite part of a given infinite array u.

To be more precise, we introduce some notations. According to a specific total order $\leq_T$ over $\mathbf{Z}_0^N$, we arrange the points $\underline{j} \in \mathbf{Z}_0^N$ so that we have $\mathbf{Z}_0^N = \{\underline{j}^{(l)}, l \in \mathbf{Z}_0 \mid \underline{j}^{(l)} <_T \underline{j}^{(l+1)}, l \in \mathbf{Z}_0\}$, and a partial

array $u^{\underline{i}} := (u_{\underline{j}})$, $\underline{j} <_T \underline{i}$ for any point $\underline{i} \in \mathbf{Z}_0^N$. For a point $\underline{i} = \underline{i}^{(l)}$, we call $\underline{i}^{(l+1)}$ the next point of $\underline{i}$, and denote it as $\underline{i} \oplus 1$. If a polynomial $f = \sum_{\underline{k} \in \mathrm{Supp}(f)} f_{\underline{k}} \underline{x}^{\underline{k}}$ $(\in R)$ satisfies

$$f[u]_{\underline{j}} := \sum_{\underline{k} \in \mathrm{Supp}(f)} f_{\underline{k}} u_{\underline{k}+\underline{j}-\underline{d}} = 0, \quad \underline{d} = \deg(f) \leq_P \underline{j} <_T \underline{i}, \tag{4}$$

we say that f is *valid* (w.r.t. u) *before* $\underline{i}$. From now on we often omit the phrase "w.r.t. u." Furthermore, if the condition (4) holds and $f[u]_{\underline{i}} \neq 0$, then we say that f is *not valid at the point* $\underline{i}$ *for the first time*. A monic (i.e. $\mathrm{hc}(f) = 1$) polynomial f which is valid before $\underline{i}$ and whose degree $\deg(f)$ is minimal w.r.t. the partial order $\leq_P$ is called a *minimal polynomial of the partial array* $u^{\underline{i}}$. Since there exist in general plural minimal polynomials f with distinct degrees $\deg(f)$ of a given partial array $u^{\underline{i}}$, we can define a minimal polynomial set $F(\underline{i})$ (or simply, F) of $u^{\underline{i}}$ associated with a finite set of points $D(\underline{i}) = \{\deg(f) \mid f \in F(\underline{i})\} \subset \mathbf{Z}_0^N$ s.t. there is a single element $f \in F(\underline{i})$ with $\deg(f) = \underline{d}$ for each $\underline{d} \in D(\underline{i})$ and there exists no polynomial h with $\deg(h) <_P \underline{d} \in D(\underline{i})$ which is valid (w.r.t. u) before $\underline{i}$. As a consequence we have a pair of subsets $\subset \mathbf{Z}_0^N$ as follows, where $\Sigma_{\underline{d}} := \{\underline{j} \in \mathbf{Z}_0^N \mid \underline{j} \geq_P \underline{d}\}$:

$$\Sigma(\underline{i}) := \cup_{\underline{d} \in D(\underline{i})} \Sigma_{\underline{d}}$$
$$\Delta(\underline{i}) := \mathbf{Z}_0^N \setminus \Sigma(\underline{i})$$

In addition to $D(\underline{i})$, letting $\Gamma_{\underline{c}} := \{\underline{j} \in \mathbf{Z}_0^N \mid \underline{j} \leq_P \underline{c}\}$ for $\underline{c} \in \mathbf{Z}_0^N$, we have a finite subset $C(\underline{i})(\subset \mathbf{Z}_0^N)$ s.t. $\Delta(\underline{i}) = \cup_{\underline{c} \in C(\underline{i})} \Gamma_{\underline{c}}$. We call $\Delta(\underline{i})$ the *delta-set* of $F(\underline{i})$, which is, roughly speaking, in form of a stack of multidimensional building blocks and whose apices are $\underline{c} \in C(\underline{i})$. As above-mentioned, there exists no polynomial h with $\deg(h) \in \Delta(\underline{i})$ which is valid before $\underline{i}$. These subsets $D(\underline{i})$, $C(\underline{i})$, $\Sigma(\underline{i})$ and $\Delta(\underline{i})$ are unique for the given array $u^{\underline{i}}$, but a minimal polynomial set $F(\underline{i})$ is not necessarily unique for $u^{\underline{i}}$. In view of the definition of minimal polynomial set $F(\underline{i})$, $\Delta(\underline{i}) \subseteq \Delta(\underline{i} \oplus 1)$. Similar notations can be used for an infinite array u, e.g. a minimal polynomial set $F(\subset R)$ of u, the delta-set $\Delta(\subset \mathbf{Z}_0^N)$ of u, etc.

The multidimensional Berlekamp-Massey algorithm (alias Berlekamp-Massey-Sakata algorithm or BMS algorithm) is just to find a minimal polynomial set $F(\underline{i})$ of a given partial array $u^{\underline{i}}$ for a fixed point $\underline{i} \in \mathbf{Z}_0^N$. Starting with the origin $\underline{0}$, we proceed to find a minimal polynomial set $F(\underline{j})$ of the partial array $u^{\underline{j}}$ iteratively at each point $\underline{j} \leq_T \underline{i}$ accordingly to the total order $\leq_T$. If $f \in F(\underline{j})$ is valid still at $\underline{j} \oplus 1$, then $f \in F(\underline{j} \oplus 1)$. However, if some $f \in F(\underline{j})$ is not valid at $\underline{j}$, then we must update these non-valid f. Whether $\Delta(\underline{i} \oplus 1) = \Delta(\underline{i})$ or not depends on certain relations among $\underline{i}$, $D(\underline{i})$ and $C(\underline{i})$. The following basic lemma [2] describing this fact stipulates the main procedure of BMS algorithm.

Lemma 1 *If a polynomial f is not valid (w.r.t. u) for the first time at $\underline{i}$, i.e.*

$$f[u]_{\underline{j}} := 0,\ \underline{d} = \deg(f) \leq \underline{j} <_T \underline{i};\ f[u]_{\underline{i}} \neq 0,$$

then there exists no polynomial g with $\deg(g) = \underline{d}' \leq_P \underline{i} - \underline{d}$ satisfying the following condition:

$$g[u]_{\underline{j}} := 0,\ \underline{d}' \leq \underline{j} \leq \underline{i}$$

Based on Lemma 1 we define the *discrepancy*, *order* and *span* of f, respectively, as

$$\mathrm{dis}(f) := f[u]_{\underline{i}} (\neq 0)$$
$$\mathrm{ord}(f) := \underline{i}, \quad \mathrm{span}(f) := \underline{i} - \underline{d}$$

Associated with the finite subset $C(\underline{i})$ related to the delta-set $\Delta(\underline{i})$, we have a finite set of polynomials $G(\underline{i}) := \{g \mid \mathrm{span}(g) \in C(\underline{i})\}$, which we call an *auxiliary polynomial set* of $u^{\underline{i}}$. An auxiliary polynomial $g \in G(\underline{i})$ is characterized by the property that it has a maximal (w.r.t. the partial order $\leq_P$) $\mathrm{span}(g)$ among the polynomials s.t. $\mathrm{ord}(g) <_T \underline{i}$. If a minimal polynomial $f \in F(\underline{i})$ fails to be valid at $\underline{i}$, minimal polynomial(s) $f' \in F(\underline{i} \oplus 1)$ at $\underline{i} \oplus 1$ can be obtained by using appropriate auxiliary polynomial(s) $g \in G(\underline{i})$ (if exist) as shown in Lemma 2 or without any $g \in G(\underline{i})$. First we have in view of Lemma 1 that, if there exists a polynomial $f \in F(\underline{i})$ with $\underline{d} = \deg(f)$ which is not valid at $\underline{i}$ and $\underline{i} - \underline{d} \notin \Delta(\underline{i})$, then $\Delta(\underline{i} \oplus 1) \neq \Delta(\underline{i})$. Thus, we define $F_N := \{f \in F(\underline{i}) \mid \mathrm{ord}(f) = \underline{i}\}$, $F_{NN} := \{f \in F_N \mid \underline{i} - \underline{d} \notin \Delta(\underline{i})\}$, $D_{NN} := \{\deg(f) \mid f \in F_{NN}\}$. Furthermore, for $\underline{c} = (c_l)_{1 \leq l \leq N} (\in C(\underline{i}))$, let $\max(\underline{d}, \underline{i} - \underline{c}) := (\max\{d_l, i_l - c_l\})_{1 \leq l \leq N}$ $(\in \mathbf{Z}_0^N)$, and let D' be the set of minimal elements $\underline{d}'$ in $D'' :=$

$\{\underline{d}' := \max(\underline{d}, \underline{i} - \underline{c}) \mid \underline{d} \in D_{NN}, \underline{c} \in C(\underline{i})\}$ ($\subset \mathbf{Z}_0^N$), and let $\hat{D}$ be the set of minimal elements in $\Sigma(\underline{i}) \setminus \Gamma_{\underline{i}}$. Now we have the following lemma about how to update F [2].

Lemma 2 *(1) For $f \in F_N \setminus F_{NN}$, there exists $\underline{c} \in C(\underline{i})$ s.t. $\underline{d} \geq_P \underline{i} - \underline{c}$. In this case, by using an auxiliary polynomial $g \in G(\underline{i})$ s.t. $\mathrm{span}(g) = \underline{c}$, we obtain*

$$h := f - \frac{\mathrm{dis}(f)}{\mathrm{dis}(g)} \underline{x}^{\underline{d}-(\underline{i}-\underline{c})} g \in F(\underline{i} \oplus 1).$$

(2) For a pair $(f, g) \in F_{NN} \times G(\underline{i})$ with $\underline{d} = \deg(f)$, $\underline{c} = \mathrm{span}(g)$, respectively, if it holds that $\underline{d}' := \max(\underline{d}, \underline{i} - \underline{c}) \in D'$, then we obtain

$$h := \underline{x}^{(\underline{i}-\underline{c})-\underline{d}} f - \tfrac{\mathrm{dis}(f)}{\mathrm{dis}(g)} g \in F(\underline{i} \oplus 1).$$

(3) For $\hat{\underline{d}} \in \hat{D}$, if there exists no $\underline{d}' \in D'$ s.t. $\hat{\underline{d}} \geq_P \underline{d}'$, then, by using $f \in F_{NN}$ s.t. $\hat{\underline{d}} \geq_P \underline{d} = \deg(f)$, we obtain

$$h := \underline{x}^{\hat{\underline{d}}-\underline{d}} f \in F(\underline{i} \oplus 1).$$

Based on the above observations we have the following form of the BMS algorithm, where some notational simplicities are used, i.e. minimal polynomial set $F(\underline{j})$ and auxiliary polynomial set $G(\underline{j})$ at each point $\underline{j}$ are denoted simply as F and G, respectively, and for $f \in F$, $g \in G$, let $\underline{d} := \deg(f)$, $\underline{c} := \mathrm{span}(g)$, $d_f := \mathrm{dis}(f)$, $d_g := \mathrm{dis}(g)$, etc. A simple result of its computation is shown in Appendix **A**, and for the purpose of comparison, the Berlekamp-Massey (BM) algorithm (the case of $N = 1$) is shown together with its simple result of its computation in Appendix **B**.

Algorithm 2 *BMS algorithm*

Step 1 (initialization): $\underline{j} := \underline{0}$; $F := \{1\}$; $D := \{\underline{0}\}$;
$G := \emptyset$; $C := \emptyset$;

Step 2 (discrepancy): for each $f \in F$ $d_f := f[u]_{\underline{j}}$;
$F_N := \{f \in F \mid d_f \neq 0\}$;
$F_{NN} := \{f \in F_N \mid \nexists \underline{c} \in C \text{ s.t. } \underline{d} \geq_P \underline{j} - \underline{c}\}$;
$D_{NN} := \{\underline{d} = \deg(f) \in D \mid f \in F_{NN}\}$;
$D'' := \{\max(\underline{d}, \underline{j} - \underline{c}) \mid \underline{d} \in D_{NN}, \underline{c} \in C\}$;
$D' := \{\text{minimal } \underline{d}' \in D''\}$;
$\hat{D} := \{\text{minimal } \hat{\underline{d}} \in \Sigma(\underline{i}) \setminus \Gamma_{\underline{i}}\}$;

Step 3 (updating): (1) for each $f \in F_N \setminus F_{NN}$
begin $h := f - \frac{d_f}{d_g} \underline{x}^{\underline{d}-(\underline{j}-\underline{c})} g$
(for $g \in G$ *s.t.* $\underline{d} \geq_P \underline{j} - \underline{c}$*);*
$F' := F \cup \{h\}$*; end;*
for each $(f, g) \in F_{NN} \times G$
s.t. $\underline{d}' := \max(\underline{d}, \underline{j} - \underline{c}) \in D'$
begin $h := \underline{x}^{\underline{d}'-\underline{d}} f - \frac{d_f}{d_g} g$;
$F' := F \cup \{h\}$*; end:*
for each $\hat{\underline{d}} \in \hat{D}$
if $\nexists \underline{d}' \in D'$ *s.t.* $\hat{d} \geq_P \underline{d}'$ *then*
for $f \in F_{NN}$ *s.t.* $\underline{d} \leq_P \hat{d}$
begin $h := \underline{x}^{\underline{d}'-\underline{d}} f$;
$F' := F \cup \{h\}$*; end:*
(2) $F := F' \setminus F_N$;
$G'' := \{g \in G \mid \exists f \in F_{NN}$
s.t. $\underline{c} <_P \underline{j} - \underline{d}\}$;
$G := (G \cup F_{NN}) \setminus G''$;
$D := \{\deg(f) \mid f \in F' \setminus F_{NN}\}$;
$C := (C \cup \{\underline{j} - \underline{d} \mid \exists f \in F_{NN}$
s.t. $\underline{j} - \underline{d} >_P \underline{c}\})$
$\setminus \{\underline{c} \in C \mid \exists f \in F_{NN}$
s.t. $\underline{j} - \underline{d} >_P \underline{c}\}$;

Step 4 (termination): $\underline{j} := \underline{j} \oplus 1$; *if* $\underline{j} <_T \underline{i}$
then go to Step 2 else stop.

A minimal polynomial set $F(\underline{i})$ is not necessarily unique for the given array u. Let $\mathcal{F}$ be the class of all reduced minimal polynomial sets $F = F(\underline{i})$ of $u^{\underline{i}}$, where $F \in \mathcal{F}$ is said to be *reduced* iff any $f \in F$ has support $\mathrm{Supp}(f)$ s.t. $\overline{\mathrm{Supp}}(f) := \mathrm{Supp}(f) \setminus \{\deg(f)\}$ is contained in the delta-set $\Delta := \Delta(\underline{i})$. Let $\deg(f) + \Delta := \{\deg + \underline{j} \mid \underline{j} \in \Delta\}$. Now, we have the following lemma [2] about the uniqueness of $F(\underline{i})$.

Lemma 3 *(Uniqueness) Let $F \in \mathcal{F}$. Then, we have that $\#\mathcal{F} = 1$ iff*

$$\bigcup_{f \in F} (\deg(f) + \Delta) \subseteq \Sigma^{\underline{i}},$$

or in other words,

$$\max_T\{\deg(f) + \mathrm{ord}(g) - \deg(g) \mid f \in F,\, g \in G\} <_T \underline{i},$$

where G is an auxiliary polynomial set of $u^{\underline{i}}$, and $\max_T\{\cdots\}$ is the maximum (w.r.t. $\leq_T$) element of the set $\{\cdots\}$.

Corollary 1 *Let $P = \{\sum_{1\leq k\leq N} c_k \underline{i}^{(k)} \in \mathbf{Z}_0^N \mid 0 \leq c_k \leq 1,\, c_k \in \mathbf{Q},\, 1 \leq k \leq N\}$ be a fundamental period parallelotope $P \subset \mathbf{Z}_0^N$ of an infinite N-D periodic array u s.t. $u_{\underline{j}+\underline{i}^{(k)}} = u_{\underline{j}}$ for any $\underline{j} \in \mathbf{Z}_0^N$, and let $2P := \{\underline{i} + \underline{j} \mid \underline{i}, \underline{j} \in P\}$. Then, If the subset $\Sigma^{\underline{i}} := \{\underline{j} \in \mathbf{Z}_0^N \mid \underline{j} <_T \underline{i}\}$ contains $2P := \{\underline{i} + \underline{j} \mid \underline{i}, \underline{j} \in P\}$, a minimal polynomial set F of a part $u^{\underline{i}}$ is a Gröbner basis of u.*

In real applications of the BMS algorithm, we usually know in advance how large the delta-set of the Gröbner basis F. In such cases we can terminate the iterations of the BMS algorithm much earlier than described in Corollary 1. We assume the total order $\leq_T$ with the weight $\underline{w} = (w_l)_{1\leq l\leq N}$ whose components w_l are almost equal to each other, i.e. $w_1 \sim w_2 \sim \cdots w_N$. This is just the case that Buchberger algorithm has the least complexity. Let $\delta := \#\Delta$ for the delta-set Δ of the Gröbner basis which is the minimal polynomial set F at the termination point, and let $\mu := \#F$ ($\sim \#G$). Then, if the computational complexity of the BMS algorithm is measured as the total number of arithmetic operations over the fixed finite field $\mathbf{F}_q$, it is $\mathcal{O}(\mu\delta^2)$ $\sim \delta^{3-\frac{1}{N}}$ since $\mu \sim \delta^{1-\frac{1}{N}}$. For example, the BMS is applied to solve a N-variate interpolation problem with m points, its computational complexity is $m^{3-\frac{1}{N}}$ because $\delta \sim m$.

We should remark that we can have various modifications of the original BMS algorithm and that the computational complexities of these versions are reduced considerably when they are applied to various practical problems including decoding of algebraic error-correcting codes because they cleverly can make use of the structures or properties of the given input data (i.e. arrays) which depend on each individual problem [6][7][5].

3 Conclusion

We have discussed that the BMS algorithm is related to Gröbner basis via multidimensional arrays and multidimensional linear recurrences satisfied by them, and that it can solve just the inverse problem of that of Buchberger algorithm, and described the essential part of the BMS algorithm. In addition, we have given some of its properties, e.g. its computational complexity, etc.

References

[1] S. Sakata, "Finding a minimal set of linear recurring relations capable of generating a given finite two-dimensional array," *J. Symbol. Comp.*, Vol.5, pp.321–337, 1988.

[2] S. Sakata, "Extension of the Berlekamp-Massey algorithm to N dimensions," *Inform. & Comp.*, Vol.84, pp.207–239, 1990.

[3] E. Berlekamp, *Algebraic Coding Theory*, McGraw-Hill: New York, 1968.

[4] J. Massey, "Shift-register synthesis and BCH decoding," *IEEE Trans. Inform. Theory*, Vol.15, pp.122–127, 1969.

[5] S. Sakata, "Applications of the BMS Algorithm to decoding of algebraic codes," preprint for Workshop on Gröbner bases in Cryptography, Coding Theory and Algebraic combinatorics, Linz, April 30 – May 5, 2006.

[6] S. Sakata, H.E. Jensen, T. Høholdt, "Generalized Berlekamp-Massey decoding of algebraic geometric codes up to half the Feng-Rao bound," *IEEE Trans. Inform. Theory*, Vol.41,

[7] S. Sakata, "N-dimensional Berlekamp-Massey algorithm for multiple arrays and construction of multivariate polynomials with preassigned zeros," in *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes: Proc. AAECC-6 (Ed. T. Mora)*, Springer: Berlin, pp.356–376, 1989.

Appendix A: BMS algorithm

[Example of computation] In Table 4 is shown a result of computations by BMS algorithm applied to

the 2-D array shown in Table 3. $\star$ implies an updated polynomial. We can make sure by Buchberger criterion that the minimal polynomial set obtained at the final iteration is a Gröbner basis. (Remark: A minimal polynomial set is not necessarily a Gröbner basis.)

Appendix B: BM algorithm

Some notational simplicities are used, i.e. a minimal polynomial and an auxiliary polynomial of each u^j are denoted simply as f and g, respectively, and let $d := \deg(f)$, $c := \mathrm{span}(g)$, $d_f := \mathrm{dis}(f)$, $d_g := \mathrm{dis}(g)$.

Algorithm 3 *BM algorithm*

Step 1 (initialization): $j := 0$; $f := 1$; $d := 0$;

$g := 0$; $c := -1$; $d_g := 0$;

Step 2 (discrepancy): $d_f := f[u]_j$;

Step 3 (updating): if $d_f \neq 0$ *then*

if $d \geq j - c$ *then*

$f := f - \frac{d_f}{d_g} x^{d-(j-c)} g$;

else begin $d' := j - c$; $h := x^{d'-d} f - \frac{d_f}{d_g} g$;

$g := f$; $c := j - d$; $d_g := d_f$; $f := h$;

$d := d'$; *end;*

Step 4 (termination): $j := j + 1$;

if $j < i$ *then go to Step 2 else stop.*

[Example of computation] In Table 6 is shown a result of computations by BM algorithm applied to the 1-D array shown in Table 5.

Table 3: 2-D array over $\mathbf{F}_2$: $u = (u_{ij})$

$i \setminus j$	0	1	2	3	4
0	0	1	0	0	1
1	1	1	1	1	
2	1	0	0		
3	1	0			
4	0	$*$			
5	1				

Table 4: computations by BMS algorithm

j	F	D	G	C
$(0,0)$	1	$(0,0)$	$-$	$-$
$(1,0)$	$\cdot$		$\cdot$	
$(0,1)$	$\star x^2$	$(2,0)$	$\star 1$	$(1,0)$
	y	$(0,1)$		
$(2,0)$	x^2	$(2,0)$	1	$(1,0)$
	$\star y + x$	$(0,1)$		
$(1,1)$	$\star x^2 + x$	$(2,0)$	1	$(1,0)$
	$y + x$	$(0,1)$		
$(0,2)$	$\cdot$		$\cdot$	
$(3,0)$	$x^2 + x$	$(2,0)$	1	$(1,0)$
	$\star xy + x^2$	$(1,1)$	$\star y + x$	$(0,1)$
	$\star y^2 + xy + x$	$(0,2)$		
$(2,1)$	$\cdot$		$\cdot$	
$(1,2)$	$x^2 + y$	$(2,0)$	1	$(1,0)$
	$\star xy + x^2 + 1$	$(1,1)$	$y + x$	$(0,1)$
	$y^2 + xy + x$	$(0,2)$		
$(0,3)$	$\cdot$		$\cdot$	
$(4,0)$	$\cdot$		$\cdot$	
$(3,1)$	$\cdot$		$\cdot$	
$(2,2)$	$\star x^3 + xy + y + x$	$(3,0)$	$\star xy + x^2 + 1$	$(2,0)$
	$\star x^2 y + x^2 + x + 1$	$(2,1)$	$\star x^2 + y$	$(1,1)$
	$y^2 + xy + x$	$(0,2)$	$y + x$	$(0,1)$
$(1,3)$	$\cdot$		$\cdot$	
$(0,4)$	$x^3 + xy + y + x$	$(3,0)$	$xy + x^2 + 1$	$(2,0)$
	$x^2 y + x^2 + x + 1$	$(2,1)$	$x^2 + y$	$(1,1)$
	$\star y^2 + x^2 + x + 1$	$(0,2)$	$y + x$	$(0,1)$
$(5,0)$	$\cdot$		$\cdot$	
$(4,1)$	$*$			

Table 5: 1-D array over $\mathbf{F}_2$: $u = (u_j)$

j	0	1	2	3	4	5	6	7
u_j	1	0	1	1	1	0	0	1

Table 6: computations by BM algorithm

j	d_f	f	d	g	c
0	1	1	0	0	-1
1	0	x	1	1	0
2	1	x	1	1	0
3	1	$x^2 + 1$	2	x	1
4	1	$x^2 + x + 1$	2	x	1
5	1	$x^3 + x^2$	3	$x^2 + x + 1$	2
6	0	$x^3 + x + 1$	3	$x^2 + x + 1$	2
7	0	$x^3 + x + 1$	3	$x^2 + x + 1$	2
8	$*$	$x^3 + x + 1$	3	$x^2 + x + 1$	2