An Interpolation Algorithm for List Decoding of Reed-Solomon Codes

Kwankyu Lee
Department of Mathematics
San Diego State University
San Diego, USA
Email: kwankyu@sogang.ac.kr

Michael E. O'Sullivan
Department of Mathematics
San Diego State University
San Diego, USA
Email: mosulliv@math.sdsu.edu

***Abstract*— The interpolation step of Sudan's list decoding of Reed-Solomon codes sets forth the problem of finding the minimal polynomial of the ideal of interpolating polynomials with respect to a certain monomial order. An efficient algorithm that solves the problem is presented based on the theory of Gröbner bases of modules. In a special case, this algorithm is shown to be equivalent with the Berlekamp-Massey algorithm for decoding Reed-Solomon codes.**

I. Introduction

Interpreting the key equation of Welch and Berlekamp [1] as a problem of finding an algebraic plane curve interpolating points with a certain degree condition, Sudan [2] developed his list decoding of Reed-Solomon codes. Soon afterward, using the concept of multiplicity at a point on an algebraic curve, Guruswami and Sudan [3] improved Sudan's list decoding so that it is capable of correcting more errors than conventional decoding algorithms for all rates of Reed-Solomon codes. Sudan's list decoding sets forth two problems: an interpolation problem and a root-finding problem. Since the interpolation problem can be solved by finding a solution of a system of linear equations, Sudan simply asserted the existence of a polynomial time algorithm solving the interpolation problem. He left it open to search for an efficient interpolation algorithm. For the root-finding problem, he cited the existence of polynomial time factorization algorithms of multivariate polynomials.

Several authors, including [4], [5], [6], and [7], formulated the interpolation problem as a problem of finding the minimal polynomial of the ideal of polynomials interpolating certain points, with respect to a monomial order. Typically an algorithm developed in this perspective is *incremental on points* in the sense that the algorithm builds a Gröbner basis of the ideal for points $\{P_1, P_2, \ldots, P_n\}$ with multiplicity m at each point by recursively computing a Gröbner basis of the ideal for $\{P_1, \ldots, P_i\}$ while i increases from 1 to n. In this paper, we also use the Gröbner basis perspective, but we employ a different strategy. We start with a set of generators of the module derived from the ideal for $\{P_1, P_2, \ldots, P_n\}$ and convert the generators to a Gröbner basis of the module, in which the minimal polynomial is found. This results in an efficient algorithm solving the interpolation problem.

II. Reed-Solomon Codes

Let $\mathbb{F}$ denote a field, fixed throughout. Although the application that concerns us is list decoding of Reed-Solomon codes over a finite field, the results of this paper are valid over an arbitrary field. Let $\mathbb{F}[x]$ be the ring of polynomials in a variable x over $\mathbb{F}$. We denote by $\mathbb{F}[x]_s$ the set of polynomials with degree $< s$, which is an s-dimensional subspace of $\mathbb{F}[x]$ as $\mathbb{F}$-vector spaces.

We fix n distinct elements $\alpha_1, \alpha_2, \ldots, \alpha_n$ from $\mathbb{F}$. Note that the evaluation map $\mathrm{ev} : \mathbb{F}[x]_n \to \mathbb{F}^n$ defined by

$$f \mapsto (f(\alpha_1), f(\alpha_2), \ldots, f(\alpha_n))$$

is an isomorphism of $\mathbb{F}$-vector spaces. The inverse map ev^{-1} is given by Lagrange interpolation as follows. Define

$$\tilde{h}_i = \prod_{j=1, j\neq i}^{n} (x - \alpha_j), \text{ and } h_i = \tilde{h}_i(\alpha_i)^{-1}\tilde{h}_i \tag{1}$$

so that $h_i(\alpha_j) = 1$ if $j = i$, and 0 otherwise. Clearly $h_1, h_2, \ldots, h_n$ form a basis of $\mathbb{F}[x]_n$. Now for a vector $v = (v_1, v_2, \ldots, v_n) \in \mathbb{F}^n$, we write

$$h_v = \mathrm{ev}^{-1}(v) = \sum_{i=1}^{n} v_i h_i \in \mathbb{F}[x]_n.$$

Let $k < n$. The *Reed-Solomon code* $\mathrm{RS}(n, k)$ over $\mathbb{F}$ is defined as the image of $\mathbb{F}[x]_k$ by ev. Since $\mathbb{F}[x]_k$ is a k-dimensional subspace of $\mathbb{F}[x]_n$, it follows that $\mathrm{RS}(n, k)$ is an $[n, k]$ linear code over $\mathbb{F}$. The minimum distance of the code meets the Singleton bound of $n - k + 1$. A generator matrix of $\mathrm{RS}(n, k)$ is

$$G = \begin{bmatrix} 1 & 1 & \ldots & 1 \\ \alpha_1 & \alpha_2 & \ldots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \ldots & \alpha_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{k-1} & \alpha_2^{k-1} & \ldots & \alpha_n^{k-1} \end{bmatrix} \tag{2}$$

from the natural basis $\{1, x, x^2, \ldots, x^{k-1}\}$ of $\mathbb{F}[x]_k$.

A parity check matrix of $\mathrm{RS}(n,k)$ is

$$H = \begin{bmatrix} u_1 & u_2 & \cdots & u_n \\ u_1\alpha_1 & u_2\alpha_2 & \cdots & u_n\alpha_n \\ u_1\alpha_1^2 & u_2\alpha_2^2 & \cdots & u_n\alpha_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ u_1\alpha_1^{n-k-1} & u_2\alpha_2^{n-k-1} & \cdots & u_n\alpha_n^{n-k-1} \end{bmatrix}, \quad (3)$$

where

$$u_i = \tilde{h}_i(\alpha_i)^{-1} = \prod_{j=1, j\neq i}^{n} (\alpha_i - \alpha_j)^{-1}. \quad (4)$$

For Sudan's list decoding of the Reed-Solomon code $\mathrm{RS}(n,k)$, the following encoding method is appropriate. A message $\omega = (\omega_0, \omega_1, \ldots, \omega_{k-1}) \in \mathbb{F}^k$ is encoded to the codeword $\omega G = \mathrm{ev}(f_\omega) \in \mathbb{F}^n$, where f_ω is the message polynomial $\omega_{k-1}x^{k-1} + \cdots + \omega_1 x + \omega_0$.

III. Sudan's List Decoding

Let $\mathbb{F}[x,y]$ be the ring of polynomials in variables x and y over $\mathbb{F}$. For $f \in \mathbb{F}[x,y]$ and $u \geq 1$, we denote by $\deg_u(f)$ the $(1,u)$-*weighted degree* of f. That is, variables x and y are assigned weights 1 and u respectively; for a monomial x^iy^j, we define $\deg_u(x^iy^j) = i+uj$; and for a polynomial f, we define $\deg_u(f)$ as the maximal $\deg_u(x^iy^j)$ for monomials x^iy^j occurring in f.

The ring $\mathbb{F}[x,y]$ may also be seen as ring of polynomials in y over $\mathbb{F}[x]$. For $f \in \mathbb{F}[x,y]$, we denote by y-$\deg(f)$ the degree of f as a polynomial in y over $\mathbb{F}[x]$.

The *multiplicity* of $f \in \mathbb{F}[x,y]$ at the origin is defined to be the smallest m such that a monomial of total degree m occurs in the polynomial f. The *multiplicity* of f at an arbitrary point $P = (a,b)$ is defined as the multiplicity of f_P at the origin, where $f_P = f(x+a, y+b)$.

Suppose that some codeword of $\mathrm{RS}(n,k)$ was sent through a noisy channel, and the vector $v \in \mathbb{F}^n$ is received by hard-decision on the channel ouput. For each $1 \leq i \leq n$, let P_i denote the point (α_i, v_i) in the plane $\mathbb{F}^2$. Now for $m \geq 1$, define

$$I_{v,m} = \{0\} \cup \{f \in \mathbb{F}[x,y] \mid f \text{ has multiplicity } m \text{ at } P_i \text{ for } 1 \leq i \leq n\},$$

which is an ideal of $\mathbb{F}[x,y]$. Sudan's list decoding is based on the following fundamental result.

Proposition 1: Let $v \in \mathbb{F}^n$ be the received vector. Suppose that $f \in I_{v,m}$ is nonzero. Let $w = \deg_{k-1}(f)$. If c is a codeword of $\mathrm{RS}(n,k)$ satisfying

$$\mathrm{wt}(v-c) < n - \frac{w}{m},$$

then h_c is a root of f as a polynomial in y over $\mathbb{F}[x]$.

It is clear from the proposition that to get the maximum decoding radius, for fixed m, one should minimize w. Thus one should choose a polynomial in $I_{v,m}$ having the smallest $(1,k-1)$-weighted degree. Having the same weighted degree, the one with smaller degree in y is preferred because this reduces the work of the root-finding step. We are thus led to consider the $>_{k-1}$ order on monomials of $\mathbb{F}[x,y]$ defined by $x^{i_1}y^{j_1} >_{k-1} x^{i_2}y^{j_2}$ when $\deg_{k-1}(x^{i_1}y^{j_1}) > \deg_{k-1}(x^{i_2}y^{j_2})$ or if $\deg_{k-1}(x^{i_1}y^{j_1}) = \deg_{k-1}(x^{i_2}y^{j_2})$ and $j_1 > j_2$. We will consider Gröbner bases of ideals of $\mathbb{F}[x,y]$ with respect to this order.

For an ideal I of $\mathbb{F}[x,y]$, the monic polynomial in I with the smallest leading term with respect to a monomial order $>$ is called the *minimal polynomial* of I with respect to $>$. Let Q be the minimal polynomial of $I_{v,m}$ with respect to $>_{k-1}$. Then Q has the smallest $(1,k-1)$-weighted degree of the polynomials in $I_{v,m}$. Moreover, Q has the smallest y-degree of those polynomials in $I_{v,m}$ with the same $(1,k-1)$-weighted degree as Q. Therefore Q is an optimal choice for Sudan's list decoding. The final result of this section gives upper bounds on $\deg_{k-1}(Q)$ and y-$\deg(Q)$.

Proposition 2: Let w and l be the values determined by

$$w = (k-1)\tilde{l} + \left\lceil \frac{N - S(\tilde{l}-1)}{\tilde{l}+1} \right\rceil - 1, \quad l = \tilde{l} - o,$$

where

$$\tilde{l} = \left\lceil \sqrt{\frac{2N}{k-1} + \frac{1}{4}} - \frac{1}{2} \right\rceil - 1, S(i) = \frac{(k-1)(i+1)(i+2)}{2},$$

and $o = 0$ if $\tilde{l} < N - S(\tilde{l}-1)$, and $o = 1$ otherwise. Then $\deg_{k-1}(Q) \leq w$ and y-$\deg(Q) \leq l$.

Let w and l be the values determined as in the proposition. Let

$$\tau = \left\lceil n - \frac{w}{m} \right\rceil - 1.$$

We conclude that for every codeword c of $\mathrm{RS}(n,k)$ satisfying $\mathrm{wt}(v-c) \leq \tau$, h_c is a root of Q as a polynomial in y over $\mathbb{F}[x]$, and there are at most l such codewords.

IV. Gröbner Basis Perspective

One way to obtain Q is to compute a Gröbner basis of the ideal $I_{v,m}$ with respect to $>_{k-1}$ and then take the minimal element of the Gröbner basis. However, computing a Gröbner basis of an ideal is generally a task of high complexity. We overcome this difficulty by using the theory of Gröbner bases of modules.

Let l be a positive integer. Let

$$\mathbb{F}[x,y]_l = \{f \in \mathbb{F}[x,y] \mid y\text{-}\deg(f) \leq l\}.$$

We view $\mathbb{F}[x,y]_l$ as a free module over $\mathbb{F}[x]$ with a free basis $1, y, y^2, \ldots, y^l$. With this basis, we may identify $\mathbb{F}[x,y]_l$ with $\mathbb{F}[x]^{l+1}$. Monomials of the module $\mathbb{F}[x,y]_l$ consist of x^iy^j with $i \geq 0$ and $0 \leq j \leq l$. A monomial order $>$ on the ring $\mathbb{F}[x,y]$ naturally induces a monomial order on the module $\mathbb{F}[x,y]_l$, which we also denote by $>$. The notions of $(1,u)$-weighted degrees and y-degrees of monomials or polynomials in $\mathbb{F}[x,y]$ also carry over to $\mathbb{F}[x,y]_l$. The minimal polynomial of a submodule of $\mathbb{F}[x,y]_l$ is defined in the same way as for an ideal of $\mathbb{F}[x,y]$.

For $l \geq 1$, we define

$$I_{v,m,l} = I_{v,m} \cap \mathbb{F}[x,y]_l.$$

Then $I_{v,m,l}$ is a submodule of $\mathbb{F}[x,y]_l$. The minimal polynomial Q of $I_{v,m}$ with respect to $>_{k-1}$ is also the minimal polynomial of $I_{v,m,l}$ with respect to $>_{k-1}$ provided that $l \geq y\text{-deg}(Q)$. In fact, we can take l as in Proposition 2. With this choice of l, we can find Q by computing a Gröbner basis of the submodule $I_{v,m,l}$ of the free module $\mathbb{F}[x,y]_l$ with respect to $>_{k-1}$. This task turns out to be much easier than that of computing a Gröbner basis of the ideal $I_{v,m}$. One reason is that there is a simple criterion for a generating set of a submodule of $\mathbb{F}[x,y]_l$ to be a Gröbner basis, given below. This criterion is the basis for an efficient algorithm computing a Gröbner basis of $I_{v,m,l}$ with respect to $>_{k-1}$.

Proposition 3: Let S be a submodule of $\mathbb{F}[x,y]_l$. Fix a monomial order $>$ on $\mathbb{F}[x,y]_l$. Suppose that $\{g_0, g_1, \ldots, g_s\}$ generates S. If the y-degrees of the leading terms of g_i for $0 \leq i \leq s$ are all distinct, then $\{g_0, g_1, \ldots, g_s\}$ is a Gröbner basis of S with respect to $>$.

The idea for the algorithm is to start with a generating set for $I_{v,m,l}$ that is relatively easy to compute, and then modify it to obtain a Gröbner basis with respect to $>_{k-1}$. The following proposition gives the desired generating set.

Proposition 4: Let $\eta = \prod_{j=1}^{n}(x-\alpha_j)$. For any $l \geq m$, $I_{v,m,l}$ is generated by $g_0, \ldots, g_l$ as an $\mathbb{F}[x]$-submodule of $\mathbb{F}[x,y]_l$ where

$$g_i = \begin{cases} (y-h_v)^i \eta^{m-i} & \text{for } 0 \leq i \leq m, \\ y^{i-m}(y-h_v)^m & \text{for } m < i \leq l. \end{cases}$$

V. An Interpolation Algorithm

Let $l \geq 1$. Let S be a submodule of $\mathbb{F}[x,y]_l$ over $\mathbb{F}[x]$. Suppose that

$$\begin{cases} g_0 = & a_{00} \\ g_1 = & a_{11}y + a_{10} \\ g_2 = & a_{22}y^2 + a_{21}y + a_{20} \\ \quad \vdots & \\ g_l = & a_{ll}y^l + \cdots + a_{l2}y^2 + a_{l1}y + a_{l0} \end{cases} \tag{5}$$

with $a_{ij} \in \mathbb{F}[x]$ are given as a set of generators of S and that $y\text{-deg}(g_i) = i$ for $0 \leq i \leq l$. Fix a monomial order $>_u$ on $\mathbb{F}[x,y]_l$. We present an algorithm computing from (5) a Gröbner basis of S with respect to $>_u$. Recall that $\{g_0, g_1, \ldots, g_l\}$ is a Gröbner basis of S if $y\text{-deg}(\text{lt}(g_i)) = i$ for each $0 \leq i \leq l$ by Proposition 3. Note that we already have $y\text{-deg}(\text{lt}(g_0)) = 0$ since $y\text{-deg}(g_0) = 0$. Our algorithm processes $g_0, g_1, \ldots, g_r$ such that $y\text{-deg}(\text{lt}(g_i)) = i$ for $0 \leq i \leq r$ while r iterates from 1 to l.

Assume that $g_0, g_1, \ldots, g_r$ satisfy

(i) $y\text{-deg}(g_i) \leq r$ for $0 \leq i \leq r$,
(ii) $y\text{-deg}(\text{lt}(g_i)) = i$ for $0 \leq i \leq r-1$, and
(iii) for every non-identity permutation $\pi = (\pi_0, \pi_1, \ldots, \pi_r)$ of $\{0, 1, \ldots, r\}$,

$$\sum_{i=0}^{r} \deg(a_{ii}) > \sum_{i=0}^{r} \deg(a_{i\pi_i}).$$

Find $s = y\text{-deg}(\text{lt}(g_r))$. If $s = r$, then we are done for $g_0, g_1, \ldots, g_r$. Suppose $s < r$. Consider

$$g_s = a_{sr}y^r + \cdots + a_{ss}y^s + \cdots,$$
$$g_r = a_{rr}y^r + \cdots + a_{rs}y^s + \cdots.$$

Note that $y\text{-deg}(\text{lt}(g_r)) = y\text{-deg}(\text{lt}(g_s)) = s$. Let

$$d = \deg(a_{rs}) - \deg(a_{ss}) \quad \text{and} \quad c = \text{lc}(a_{rs})\text{lc}(a_{ss})^{-1}.$$

We update g_s and g_r as follows. If $d \geq 0$, then set

$$g_r \leftarrow g_r - cx^d g_s.$$

If $d < 0$, then set, storing g_s in a temporary place,

$$g_s \leftarrow g_r, \quad g_r \leftarrow x^{-d}g_r - cg_s.$$

We repeat this processing on $g_0, g_1, \ldots, g_r$ until we have $y\text{-deg}(\text{lt}(g_r)) = r$.

Combined with the set of generators of $I_{v,m,l}$ given in the previous section, we obtain the following interpolation algorithm.

Interpolation Algorithm I. Given input $v = (v_1, v_2, \ldots, v_n)$ and parameters m and l, this algorithm finds the minimal polynomial of $I_{v,m,l}$ with respect to monomial order $>_{k-1}$. Throughout the algorithm we let $g_i = \sum_{j=0}^{l} a_{ij}y^j$ for $0 \leq i \leq l$.

I1. Compute $h_v = \sum_{i=1}^{n} v_i h_i$. For $0 \leq i \leq m$, set

$$g_i \leftarrow (y-h_v)^i \prod_{j=1}^{n}(x-\alpha_j)^{m-i}$$

and for $m < i \leq l$, set

$$g_i \leftarrow y^{i-m}(y-h_v)^m.$$

Set $r \leftarrow 0$.
I2. Increase r by 1. If $r \leq l$, then proceed; otherwise go to step I6.
I3. Find $s = y\text{-deg}(\text{lt}(g_r))$. If $s = r$, then go to step I2.
I4. Set $d \leftarrow \deg(a_{rs}) - \deg(a_{ss})$ and $c \leftarrow \text{lc}(a_{rs})\text{lc}(a_{ss})^{-1}$.
I5. If $d \geq 0$, then set

$$g_r \leftarrow g_r - cx^d g_s.$$

If $d < 0$, then set, storing g_s in a temporary variable,

$$g_s \leftarrow g_r, \quad g_r \leftarrow x^{-d}g_r - cg_s.$$

Go back to step I4.
I6. Let Q be the g_i with the smallest leading term. Output Q and the algorithm terminates.

VI. Classical Case

From now on, we consider Sudan's list decoding for the case $m = l = 1$. In this case, our algorithm is intimately connected with the classical decoding algorithms for Reed-Solomon codes.

Proposition 5: Let $\tau = \lfloor (n-k)/2 \rfloor$. There is at most one codeword c satisfying $\mathrm{wt}(v-c) \le \tau$. Suppose that there is such a codeword c. Let $e = v - c$, and

$$f_e = \prod_{e_i \neq 0} (x - \alpha_i).$$

Then $f_e(y - h_c)$ is the minimal polynomial of $I_{v,1,1}$ with respect to $>_{k-1}$.

Henceforth, we assume that there occurred no more than $\tau = \lfloor (n-k)/2 \rfloor$ errors to the sent codeword. Then by the proposition, the sent codeword c is the unique codeword satisfying $\mathrm{wt}(v-c) \le \tau$, and the message polynomial h_c is obtained by one division from the minimal polynomial of $I_{v,1,1}$. The Interpolation Algorithm is also substantially simplified when it is applied to $I_v = I_{v,1,1}$. We will write $g_0 = Ay + B$ and $g_1 = Cy + D$.

Decoding Algorithm D. Given the received vector $v = (v_1, v_2, \ldots, v_n)$, this algorithm finds the message polynomial h_c. The polynomials $\eta = \prod_{j=1}^{n}(x - \alpha_j)$ and h_i as in (1) for $1 \le i \le n$ are precomputed.

D1. Compute $-h_v = -\sum_{i=1}^{n} v_i h_i$.

D2. Set

$$A \leftarrow 0, B \leftarrow \eta, C \leftarrow 1, D \leftarrow -h_v.$$

D3. If $\deg(C) + k - 1 \ge \deg(D)$, then go to step D6.

D4. Set $d \leftarrow \deg(D) - \deg(B)$ and $c \leftarrow \mathrm{lc}(D)\mathrm{lc}(B)^{-1}$.

D5. If $d \ge 0$, then set

$$C \leftarrow C - cx^d A, D \leftarrow D - cx^d B.$$

If $d < 0$, then set, storing A and B in temporary variables,

$$A \leftarrow C, B \leftarrow D, C \leftarrow x^{-d}C - cA, D \leftarrow x^{-d}D - cB.$$

Go back to step D3.

D6. Set $h \leftarrow -D/C$. Output h and the algorithm terminates.

This algorithm is essentially the Euclidean algorithm. We can see this by consolidating consecutive rounds of D3, D4, D5 in which $d \ge 0$. This amounts to the following replacement steps.

E4. Compute Q and R such that $B = QD + R$, $\deg(R) < \deg(D)$ by the Euclidean algorithm.

E5. Set, storing A in a temporary variable

$$A \leftarrow C, B \leftarrow D, C \leftarrow A - QC, D \leftarrow R.$$

Go back to step D3.

The Berlekamp-Massey algorithm is also intimately related with Algorithm D. To see this, note that the condition in step D3 may be rewritten as $\deg(D) - \deg(C) \le k-1$. By keeping track of the value of $\deg(C) - \deg(D)$, we get yet another algorithm, which does the same computations as Algorithm D with a slightly different control structure.

Yet Another Algorithm Y. Given the received vector $v = (v_1, v_2, \ldots, v_n)$, this algorithm finds the message polynomial h_c. The polynomials $\eta = \prod_{j=1}^{n}(x - \alpha_j)$ and h_i as in (1) for $1 \le i \le n$ are precomputed.

Y1. Compute $-h_v = -\sum_{i=1}^{n} v_i h_i$.

Y2. Set

$$A \leftarrow 0, B \leftarrow \eta, C \leftarrow 1, D \leftarrow -h_v$$

and set $s \leftarrow n - 1$.

Y3. If $\deg(C) + s > \deg(D)$, then go to step Y6.

Y4. Set $d \leftarrow \deg(D) - \deg(B)$ and $c \leftarrow \mathrm{lc}(D)\mathrm{lc}(B)^{-1}$.

Y5. If $d \ge 0$, then set

$$C \leftarrow C - cx^d A, D \leftarrow D - cx^d B.$$

If $d < 0$, then set, storing A and B in temporary variables,

$$A \leftarrow C, B \leftarrow D, C \leftarrow x^{-d}C - cA, D \leftarrow x^{-d}D - cB.$$

Y6. Set $s \leftarrow s - 1$. If $s \ge k$, then go back to step Y3. Otherwise, proceed.

Y7. Output $-D/C$ and the algorithm terminates.

We can now compare Algorithm Y with the Berlekamp-Massey algorithm. Since $I_v = \langle \prod_{j=1}^{n}(x - \alpha_j), y - h_v \rangle$, we have

$$ay + b \in I_v \iff ah_v + b = \psi \prod_{j=1}^{n}(x - \alpha_j)$$

for a unique $\psi \in \mathbb{F}[x]$. So there is a one-to-one correspondence

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \text{ with } Ay + B \in I_v, Cy + D \in I_v \leftrightarrow \begin{bmatrix} A & \Psi \\ C & \Phi \end{bmatrix}$$

with $Ah_v + B = \Psi \prod_{j=1}^{n}(x - \alpha_j)$, $Ch_v + D = \Phi \prod_{j=1}^{n}(x - \alpha_j)$.

With this correspondence in mind, we translate each statement of Algorithm Y on the data A, B, C, D into a statement on the corresponding data A, C, Ψ, Φ. In particular, we need to extract the information about B and D from the information about A, C, Ψ, Φ.

For a polynomial f in x, let $f[x^i]$ denote the coefficient of the term x^i of f. Look at step Y3. Since $\deg(C) + s \ge \deg(D)$ holds at step Y3, the condition $\deg(C) + s > \deg(D)$ is equivalent to $D[x^{\deg(C)+s}] = 0$. Let $\mu = D[x^{\deg(C)+s}]$. If $\mu \neq 0$ so that we move on to step Y4, then $\deg(D) = \deg(C) + s$ and $\mathrm{lc}(D) = \mu$. Observe that $\deg(B)$ and $\mathrm{lc}(B)$ is set equal to $\deg(D)$ and $\mathrm{lc}(D)$ at step Y4 when we have $d < 0$. The question is how to get the value μ without D. The answer is that μ may be computed using the linear recursion that C defines on the syndromes. That is

$$\mu = -(C\sigma_v^*)[x^{p+s-k}] = -(C_p\sigma_{n-1-s} + \cdots + C_0\sigma_{n-1-s-p}).$$

where $(\sigma_0, \sigma_1, \ldots, \sigma_{n-k-1}) = vH^T$.

We use variables $q = \deg(B)$ and $\nu = \mathrm{lc}(B)$. Now we are ready to translate Algorithm Y into an equivalent

Algorithm T. Given input $v = (v_1, v_2, \ldots, v_n)$, this algorithm finds f_e and Φ satisfying (6).

T1. Compute $(\sigma_0, \sigma_1, \ldots, \sigma_{n-k-1}) = vH^T$.

T2. Set $s \leftarrow n - 1$, $p \leftarrow 0$, $q \leftarrow n$, $\nu \leftarrow 1$, and

$$A \leftarrow 0, \Psi \leftarrow 1, C \leftarrow 1, \Phi \leftarrow 0.$$

T3. Let $C = C_p x^p + \cdots + C_0$. Compute $\mu = -(C_p \sigma_{n-1-s} + \cdots + C_0 \sigma_{n-1-s-p})$.

T4. If $\mu = 0$, then go to step T6. Otherwise, set $d \leftarrow p+s-q$ and $c \leftarrow \mu/\nu$.

T5. If $d \geq 0$, then set

$$C \leftarrow C - cx^d A, \Phi \leftarrow \Phi - cx^d \Psi.$$

If $d < 0$, then set, storing A and Ψ in temporary variables,

$$A \leftarrow C, \Psi \leftarrow \Phi, C \leftarrow x^{-d} C - cA, \Phi \leftarrow x^{-d}\Phi - c\Psi$$

and, storing p in a temporary variable, $p \leftarrow q - s$, $q \leftarrow p + s$, and $\nu \leftarrow \mu$.

T6. Set $s \leftarrow s - 1$. If $s \geq k$, then go back to step T3. Otherwise, proceed.

T7. Output C and Φ, and the algorithm terminates.

In step T7, C and Φ are output instead of $-D/C$. Recall that at this point $Cy + D = f_e(y - h_c)$. Therefore $Ch_v + D = f_e(h_v - h_c) = f_e h_e$. Hence we have

$$f_e h_e = \Phi \prod_{j=1}^{n} (x - \alpha_j). \tag{6}$$

Our algorithm T is slightly different from the standard formulation of the Berlekamp-Massey algorithm. It is apparent that $p+q = n$ always holds in Algorithm T. Removing variable q using this fact and making variable changes

$$\tilde{s} = n - s, \quad \tilde{\mu} = -\mu, \quad \tilde{\nu} = -\nu, \quad \tilde{\sigma}_i = \sigma_{i-1},$$

yields the usual formulation.

Suppose that we have f_e and Φ that Algorithm T output. Let α_i be a root of f_e. Taking formal derivatives of each side of (6) and evaluating at α_i, we get the *Forney's formula*

$$e_i = f_e'(\alpha_i)^{-1} \Phi(\alpha_i) \prod_{j=1, j\neq i}^{n} (\alpha_i - \alpha_j),$$

where $'$ denotes the formal derivative.

VII. Concluding Remarks

We presented an efficient algorithm solving the interpolation problem based on the theory of Gröbner bases of modules. Since our algorithm computes a Gröbner basis of a certain submodule of a free module, we may compare our algorithm with the general algorithm computing a Gröbner basis of submodules of free modules over polynomial rings, namely Buchberger's algorithm. Indeed, after a careful comparison, it is possible to view our algorithm as a version of Buchberger's algorithm, optimized for our special submodule. Moreover, Proposition 3, on which our algorithm is based, can be viewed as an application of Buchberger's S-pair criterion. In this view, our contribution is in the optimization of Buchberger's algorithm for Sudan's list decoding of Reed-Solomon codes.

We showed that the Berlekamp-Massey algorithm can be viewed as a disguised form of the simplest case of our interpolation algorithm. The Berlekamp-Massey algorithm has been a basic model of decoding algorithms for algebraic geometric codes, which are natural extensions of Reed-Solomon codes. On the other hand, the theory of Gröbner bases is the basic tool of computational algebraic geometry. These facts make us expect that our algorithm can be naturally extended for Sudan's list decoding of algebraic geometric codes.

Acknowledgment

The first author was supported by the Korea Research Foundation Grant funded by Korea Government (MOEHRD, Basic Research Promotion Fund) (KRF-2005-214-C00009).

References

[1] L. Welch and E. Berlekamp, "Error correction for algebraic block codes," U. S. Patent 4 633 470, issued Dec. 30, 1986.

[2] M. Sudan, "Decoding of Reed-Solomon codes beyond the error-correction bound," *J. Complexity*, vol. 13, no. 1, pp. 180–193, 1997.

[3] V. Guruswami and M. Sudan, "Improved decoding of Reed-Solomon and Algebraic-Geometry codes," *IEEE Trans. Inform. Theory*, vol. 45, no. 6, pp. 1757–1767, 1999.

[4] R. R. Nielsen and T. Høholdt, "Decoding Reed-Solomon codes beyond half the minimum distance," in *Coding Theory, Cryptogrphy and related areas*, J. Buchmann, T. Høholdt, H. Stichtenoth, and H. Tapia-Recillas, Eds. Springer, 2000, pp. 221–236.

[5] H. O'Keeffe and P. Fitzpatrick, "Gröbner basis solutions of constrained interpolation problems," *Linear Algebra Appl.*, vol. 351/352, pp. 533–551, 2002.

[6] M. Alekhnovich, "Linear Diophantine equations over polynomials and soft decoding of Reed-Solomon codes," *IEEE Trans. Inform. Theory*, vol. 51, no. 7, pp. 2257–2265, 2005.

[7] J. B. Farr and S. Gao, "Gröbner bases, Padé approximation, and decoding of linear codes," in *Coding Theory and Quantum Computing*, ser. Contemp. Math. Amer. Math. Soc., 2005, vol. 381.