# A Reduced-Complexity Algorithm For Polynomial Interpolation

Yuan Zhu
Dep. of Software Eng.
Sun Yat-sen University
Guangzhou 510006, GD, China
Email: 41656955@qq.com

Siyun Tang
Dep. of Electronics & Communication Eng.
Sun Yat-sen University
Guangzhou 510006, GD, China
Email: tangsiy@mail2.sysu.edu.cn

***Abstract*—Most traditional bivariate polynomial interpolation algorithms need to construct the Gröbner basis of a module for the interpolation result. In this paper, we present an algorithm that constructs the basis for a gradually extending submodule to save computation, based on a partial order of the elements of the submodule's Gröbner basis. It also can be generalized for negative weighted interpolation and multivariate interpolation.**

## I. Introduction

Some efficient list-decoding[1][2][3] and algebraic soft-decision decoding[4][5] algorithms for RS code have been proposed in the past decades. Most of them are based on bivariate polynomial interpolation. As mentioned in [6], either in software or in hardware, most of the bivariate interpolation is carried out using the algorithm of Koetter[7] because it can be used for the re-encoding coordinate transformation[8], which reduces the complexity by orders of magnitude. The Koetter's algorithm or its re-encoding coordinate transformation version can incrementally construct a Gröbner basis for a module defined by the interpolation constraints. They will output the minimal element from the basis as the interpolation result. However, as we only need one element, some of the computation is unnecessary for final result.

In this paper, we present an improved algorithm for bivariate polynomial interpolation. It constructs the Gröbner basis of a gradually extending module. The computation reduction comes from the decrement of the size of the module. Meanwhile we can generalize it for negative weighted bivariate interpolation and multivariate interpolation.

The rest of this paper is organized as follows. In the next section, we briefly review the Koetter's interpolation algorithm and discuss the potential of computation reduction. In Section III, we develop an efficient basis construction algorithm and prove its correctness. In Section IV and Section V, negative weighted interpolation and multivariate interpolation algorithm are introduced. In Section VI, we will compare the complexity by numerical results. In Section VII, a brief discussion of the results will be concluded.

## II. Koetter's Interpolation Algorithm And the Potential Complexity Reduction

In a field $\mathbb{F}$, let $\mathbb{F}[x,y]$ denote the ring of the bivariate polynomial, $\mathbb{F}_L[x,y] \subset \mathbb{F}[x,y]$ denote the set of the polynomials with y-degree equal to or less than $L$. In fact, $\mathbb{F}_L[x,y]$ is a $\mathbb{F}[x]$-module. We define a total order $\prec$ between different bivariate monomials by comparing $(1,k)$-weighted degree: if $i_1 + kj_1 < i_2 + kj_2$, or $i_1 + kj_1 = i_2 + kj_2$ and $j_1 < j_2$, then $x^{i_1}y^{j_1} \prec x^{i_2}y^{j_2}$. By the total order, we can list every term of any polynomial $Q(x,y)$ in descending order. In such a descending order, let $LT(Q(x,y))$ denote the leading term of $Q(x,y)$, $Q(x,y) \prec Q'(x,y)$ if and only if $LT(Q(x,y)) \prec LT(Q'(x,y))$. We define that $Q(x,y)$ passes $(\alpha,\beta)$ with multiplicity $m$ if and only if the degree of all the terms of $Q(x+\alpha, y+\beta)$ is equal to or greater than $m$. The bivariate polynomial interpolation is to seek the non-zero polynomial $Q(x,y)$ in $\mathbb{F}_L[x,y]$ with the minimal $(1,k)$-weighted degree which passes the $n$ points $(x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n)$ in $\mathbb{F}^2$ with multiplicity $m_1,m_2,\ldots,m_n$. Let $D_{r,s}(Q(x,y))$ denote the $(r,s)$-th partial derivative of polynomial $Q(x,y)$. Let functional set $\{D_1,D_2,\ldots,D_C\}$ stand for the total $C$ interpolation constraints that the interpolation result has to satisfy. For functional $D_i$, we write $K_i$ as its kernel: $K_i = ker(D_i) = \{Q(x,y) \in \mathbb{F}_L[x,y] : D_i(Q(x,y)) = 0\}$. We define the cumulative kernel $\overline{K_0} = \mathbb{F}_L[x,y]$, $\overline{K_u} = \overline{K_{u-1}} \cap K_u$.

To do the $(1,k)$-weighted interpolation, Koetter's algorithm works as follows: let $G = \{1,y,\ldots,y^{L-1},y^L\}$, $G$ is the Gröbner basis of $\overline{K_0}$. Then we get the Gröbner basis of $\overline{K_u}$ by updating the elements of the Gröbner basis of $\overline{K_{u-1}}$ to satisfy the constraint $D_u$. When $u = C$, the minimal element of the Gröbner basis of $\overline{K_u}$ is the interpolation result.

We may notice that $G$ is a partially ordered set (or poset) by divisibility. When $k > 0$, the weighted degree of any quotient is positive. We denote the partial order $\prec_+$. Obviously, if $g \prec_+ h$, then $g \prec h$. Suppose we have a polynomial module $\overline{K_u}$ with Gröbner basis $G_u = \{g_0,g_1,\ldots,g_{L-1},g_L\}$ and $G_u$ is a poset partially ordered by $\prec_+$. Let $g_t, g_s \in G_u$ and $g_t \prec_+ g_s$. By the elimination property of Gröbner basis, we have following result.

**Lemma 1.** *For submodule $\overline{K_{u'}}$ of $\overline{K_u}(u' > u)$ with Gröbner basis $\{g'_0,g'_1,\ldots,g'_{L-1},g'_L\}$, if $LT(g'_t) = LT(g_t)$, then $g'_s$ has the same leading term with $g_s$.*

*Proof:* Because $g'_t(g_s/g_t) \in \overline{K_{u'}}$ and $LT(g'_t(g_s/g_t)) = LT(g_s)$, so $g'_t(g_s/g_t)$ can be the candidate of $g'_s$, which has the same leading term with $g_s$. ∎

**Corollary 2.** *If $LT(g'_t) = LT(g_t)$, then $g'_t(g_s/g_t)$ can be the candidate of $g'_s$.*

According to Corollary 2, we get a rule to reduce the computation of updating some elements in basis. For submodule $\overline{K_{u'}}$ of $\overline{K_u}(u' > u)$, the leading terms of its Gröbner basis' elements can also form a poset by $\prec_+$. If $LT(g'_s)$ is not the least element in the poset, then $g'_s$ can be directly deduced by the element with least leading term and their quotient in the original basis. So until $LT(g'_s)$ is to become the least element, we do not need to compute $g'_s$. However, if the element $g'_t$ with the $\prec_+$-least leading term is to update and change its leading term first time, we should compute other elements which will have their leading terms to be the least one in the poset. Because after $g'_t$ 's change, we lose the way to compute them in low complexity.

The following sections will describe how to use this rule in interpolation.

## III. A Reduced-Complexity Algorithm for Bivariate Polynomial Interpolation in Positive Weighted Degree

For $(1,k)$-weighted bivariate polynomial interpolation, the Gröbner basis of $\overline{K_0}$ is $G = \{g_0 = 1, g_1 = y, \ldots, g_L = y^L\}$ with elements in ascending order by $(1,k)$-weighted degree when $k > 0$. $\prec_+$ here is a total order for $G$, or a chain. If we remove the least element, the rest elements also form a chain with only one least element. So we can start to update element $g_{j+1}$ until $g_j$ changes its leading term to $xy^j$ by Corollary 2.

The reduced-complexity algorithm for bivariate polynomial interpolation in positive weighted degree is presented in Algorithm 1. In Step 1, the "update list" $G$ is initialized with only one element while Koetter's algorithm begins with $L+1$. In Step 17, an additional element is added to $G$. We will prove its correctness briefly in following content.

**Theorem 3.** *In Algorithm 1, G is the Gröbner basis of submodule $\overline{K_u} \bigcap \mathbb{F}_{|G|-1}[x,y]$.*

*Proof:* By induction, when $u = 0$, $|G| = 1$, the theorem is true. When $|G| \geq 1$, all the elements in $G$ must be correct as traditional algorithm until we add a new element. From Lemma 1, when the latest added element $g_{j^*} = f$ is to change its leading term first time in Step 19, we compute the next element by $g_{j^*+1} \leftarrow yf \in \overline{K_{u-1}}$. Then by linearity $\Delta_{j^*+1} = y_i\Delta$, $yf - (\Delta_{j^*+1}/\Delta)f = (y-y_i)f \in \overline{K_u}$. The latest added element $(y-y_i)f$ has the same leading term with $y^{|G|-1}$ and $y^{|G|-1}$ is an element of the Gröbner basis of $\overline{K_0}$. So $G$ is the Gröbner basis of $\overline{K_u} \bigcap \mathbb{F}_{|G|-1}[x,y]$ after adding $g_{j^*+1}$. ■

In fact, we generate Gröbner basis of a gradually extending submodule $\overline{K_u} \bigcap \mathbb{F}_{|G|-1}[x,y]$ in the process of Algorithm 1 rather than that of $\overline{K_u} \bigcap \mathbb{F}_L[x,y]$. So we can skip some computation of updating elements in early steps. It is possible that $|G|$ will be less than $L+1$ at the end of Algorithm 1. In this case, the output is correct because the element $g_{|G|-1}$ must have the leading term $y^{|G|-1}$, then the Gröbner basis of $\overline{K_C}$ can be $G \bigcup \{g_{|G|-1}y, g_{|G|-1}y^2, \ldots, g_{|G|-1}y^{L-(|G|-1)}\}$ , which shares the same minimal element with $G$.

**Algorithm 1** Reduced-Complexity Bivariate Interpolation Algorithm In Positive Weighted Degree

```
Input:
    Interpolation points $(x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n)$; multiplicity $m_1,m_2,\ldots,m_n$; maximum $y$-degree $L$
Output:
    A non-zero polynomial $Q(x,y) \in \mathbb{F}_L[x,y]$ satisfying the interpolation problem with minimal $(1,k)$-degree, $k > 0$
1:  $G \leftarrow \{g_0 = y^0\}$,$u \leftarrow 0$
2:  for each $i$ in $[1,n]$ do
3:    for each $(r,s)$ from $(0,0)$ to $(m_i-1,0)$ by $(m_i-1,1)$ lex order do
4:      for each $j$ in $[0,|G|-1]$ do
5:        $\Delta_j \leftarrow D_{r,s}(g_j(x_i,y_i))$
6:      end for
7:      $J \leftarrow \{j|\Delta_j \neq 0\}$, $u \leftarrow u+1$
8:      if $J \neq \emptyset$ then
9:        $j^* \leftarrow \arg\min_{j\in J}(g_j)$
10:       $f \leftarrow g_{j^*}$, $\Delta \leftarrow \Delta_{j^*}$
11:     end if
12:     for all $j \in J$ do
13:       if $j \neq j^*$ then
14:         $g_j \leftarrow g_j - \frac{\Delta_j}{\Delta}f$
15:       else
16:         if $j^* = |G|-1$ and $j^* < L$ then
17:           $g_{j^*+1} \leftarrow (y-y_i)f$, $G \leftarrow G \cup \{g_{j^*+1}\}$
18:         end if
19:         $g_{j^*} \leftarrow (x-x_i)f$
20:       end if
21:     end for
22:   end for
23: end for
24: $Q(x,y) = \arg\min_{g\in G}(g)$
```

When $L = +\infty$, $G$ is the Gröbner basis of the ideal defined by the interpolation constraints in $\mathbb{F}[x,y]$ without the limit of maximum $y$-degree. We get the basis from the affine zeros of the ideal, while Buchberger[9] gets the basis from elements in the ideal and Sakata[10] gets it as a minimal set of bivariate connection polynomials for a finite two-dimensional sequence.

Algorithm 1 can not work when $k < 0$. We will generalize it for negative weighted degree interpolation in the next section.

## IV. A Reduced-Complexity Algorithm for Bivariate Polynomial Interpolation in Negative Weighted Degree

### A. The Interpolation Algorithm for Negative Weighted Degree

Some other efficient decoding algorithms, including Wu's algorithm[2] and re-encoding coordinate transformation algorithm[8], need to do negative weighted degree polynomial interpolation. We will discuss how to simplify the interpolation in negative weighted degree.

Ordered by $(1,k)$-weighted degree, the Gröbner basis of $\overline{K_0}$ is $G = \{y^L, y^{L-1}, \ldots, y, 1\}$ with elements in ascending order when $k < 0$. We can generalize the definition of divisibility: we define polynomial $f_0(x, y)$ is divisible by polynomial $f_1(x, y)$, if the result of $f_0(x, y)/f_1(x, y)$ is a polynomial and its weighted degree is non-negative; or the result is a reduced rational polynomial, with a polynomial in non-negative weighted degree as its numerator and a polynomial in non-positive weighted degree as denominator. Under the generalization of the definition of the divisibility, $G$ is also a total order set by $\prec_+$ as the quotient of adjacent elements is $y^{-1}$ with positive weighted degree. However, $y^{-1}$ can not be derivable at the points with zero in $y$-coordinate, including the $(0, 0)$-th partial derivative. So we can not compute the new element by multiplying the quotient as Corollary 2.

But we can interpolate these points first to generate a new submodule instead of $\mathbb{F}_L[x, y]$. Let $\overline{K^{(t)}}$ denote the kernel in $\mathbb{F}_L[x, y]$ by the constraints of the $t$ interpolation points with zero in $y$-coordinate. If $t = 0$, then $\overline{K^{(t)}} = \mathbb{F}_L[x, y]$. Let $[j]^+ = \max(j, 0)$. From (37) of [8], the Gröbner basis of $\overline{K^{(t)}}$ is $\{y^L b_L(x), y^{L-1}\prod_{i=L-1}^{L} b_i(x), \ldots, y\prod_{i=1}^{L} b_i(x), \prod_{i=0}^{L} b_i(x)\}$ where

$$b_j(x) = \begin{cases} \prod_{i=1}^{t}(x - x_i)^{[m_i-j]^+-[m_i-(j+1)]^+} & t \neq 0, j < L \\ \prod_{i=1}^{t}(x - x_i)^{[m_i-L]^+} & t \neq 0, j = L \\ 1 & t = 0 \end{cases}$$

The basis is correct even $k$ is a negative number because terms of any element have the same $y$-degree. Let $a_j = \prod_{i=j}^{L} b_j(x)$. When $t > 0$, $a_j = \prod_{i=1}^{t}(x - x_i)^{[m_i-(j+1)]^+}$. So if we first interpolate the $t$ points, then we have the basis $G = \{g_L = y^L a_L, g_{L-1} = y^{L-1}a_{L-1}, \ldots, g_1 = y^1 a_1, g_0 = y^0 a_0\}$ for $\overline{K^{(t)}}$, with the quotient of adjacent elements $q_j = (y^{j-1}a_{j-1})/(y^j a_j) = b_{j-1}(x)/y$. Now $G$ is still a full order set by $\prec_+$ and $q_j$ can be derivable at the rest points. So we get Algorithm 2 to interpolate all the rest points like Algorithm 1.

**Theorem 4.** *In Algorithm 2, $G$ is the Gröbner basis of submodule $\overline{K_u} \bigcap \overline{K^{(t)}} \bigcap (\mathbb{F}_L[x, y] \setminus \mathbb{F}_{L-|G|}[x, y])$.*

*Proof:* By induction, when $u = 0$, $|G| = 1$, the theorem is true. When $|G| \geq 1$, all the elements in $G$ must be correct as traditional algorithm until we add a new element. From the conclusion in Lemma 1, when the latest added element $g_{j^*} = f$ is to change its leading term first time in Step 20, we need to compute the next element by $g_{j^*-1} \leftarrow (b_{j^*-1}(x)/y)f \in \overline{K_{u-1}}$. Then by linearity $\Delta_{j^*-1} = b_{j^*-1}(x_i)\Delta/y_i$, $(b_{j^*-1}(x)/y)f - (\Delta_{j^*-1}/\Delta)f = (b_{j^*-1}(x)/y - b_{j^*-1}(x_i)/y_i)f \in \overline{K_u}$. As the latest added element $(b_{j^*-1}(x)/y - b_{j^*-1}(x_i)/y_i)f$ has the same leading term with $y^{L-|G|+1}a_{L-|G|+1}$. $y^{L-|G|+1}a_{L-|G|+1}$ is an element of Gröbner basis of $\overline{K^{(t)}}$. So $G$ is the Gröbner basis of $\overline{K_u} \bigcap \overline{K^{(t)}} \bigcap (\mathbb{F}_L[x, y] \setminus \mathbb{F}_{L-|G|}[x, y])$ after adding $g_{j^*-1}$. ■

We have to notice that Algorithm 2 always output a polynomial. Because in Step 17, $f$ has $y$ as its factor as $g_0$ has not added into $G$ and there is no term in element of $G$ with zero $y$-degree.

**Algorithm 2** Reduced-Complexity Bivariate Interpolation Algorithm in Negative Weighted Degree

**Input:**
Interpolation points $(x_1, 0), (x_2, 0), \ldots, (x_t, 0), (x_{t+1}, y_{t+1}), (x_{t+2}, y_{t+2}), \ldots, (x_{n-1}, y_{n-1}), (x_n, y_n)$, $y_i \neq 0$, $i = t+1, t+2, \ldots, n$; multiplicity $m_1, m_2, \ldots, m_n$; maximum $y$-degree $L$

**Output:**
A non-zero polynomial $Q(x, y) \in \mathbb{F}_L[x, y]$ satisfying the interpolation problem with minimal $(1, k)$-degree, $k < 0$

```
1:  $G \leftarrow \{g_L = b_L y^L\}$, $u \leftarrow 0$
2:  for each $i$ in $[t+1, n]$ do
3:    for $(r, s)$ from $(0, 0)$ to $(m_i - 1, 0)$ by $(m_i - 1, 1)$ lex order do
4:      for each $j$ in $[L - |G| + 1, L]$ do
5:        $\Delta_j \leftarrow D_{r,s}(g_j(x_i, y_i))$
6:      end for
7:      $J \leftarrow \{j | \Delta_j \neq 0\}$, $u \leftarrow u + 1$
8:      if $J \neq \emptyset$ then
9:        $j^* \leftarrow \arg\min_{j \in J}(g_j)$
10:       $f \leftarrow g_{j^*}$, $\Delta \leftarrow \Delta_{j^*}$
11:     end if
12:     for all $j \in J$ do
13:       if $j \neq j^*$ then
14:         $g_j \leftarrow g_j - \frac{\Delta_j}{\Delta} f$
15:       else
16:         if $j^* = L - |G| + 1$ and $j^* > 0$ then
17:           $g_{j^*-1} \leftarrow (b_{j^*-1}(x)/y - b_{j^*-1}(x_i)/y_i)f$
18:           $G \leftarrow G \cup \{g_{j^*-1}\}$
19:         end if
20:         $g_{j^*} \leftarrow (x - x_i)f$
21:       end if
22:     end for
23:   end for
24: end for
25: $Q(x, y) = \arg\min_{g \in G}(g)$
```

## *B. Using Algorithm 2 to Optimize Re-encoding Coordinate Transformation Algorithm*

The re-encoding coordinate transformation algorithm[8] reduces the $(1, k)$-weighted bivariate interpolation problem from $n$ points to $n - k - 1$ when $k > 0$, which is very efficient in list-decoding of high rate RS code. After proper coordinate transformation, it also uses Koetter's interpolation algorithm to do $(1, -1)$-weighted $(x, z)$ bivariate interpolation for the transformed $n - k - 1$ points. However, we can not directly use our algorithm to delay the element's update in the process because there may exist some points in the $n - k - 1$ points with zero in $z$-coordinate.

For $k + 1$ points in different $x$-coordinate, the re-encoding algorithm gets a $k$-degree polynomial $y - e(x)$ to interpolate them using Lagrange interpolation algorithm. However, this polynomial may also pass some other points. We should do some modification here to avoid the future appearance of zero in $z$-coordinate. Suppose $y - e(x)$ passes $k'(k' > k)$ points $(x_1, y_1), (x_2, y_2), \ldots, (x_{k'}, y_{k'})$ of the $n$ interpolation points. Then we do the transformation for the rest $n - k'$ points. For point $(x_i, y_i), i > k'$, if it has different $x$-coordinate with the first $k'$ points, $(x_i, y_i)$ is transformed into $(x_i, z_i = g^{-1}(x_i)(y_i - e(x_i)))$, $g(x) = \prod_{i=1}^{k'}(x - x_i)$; else $(x_i, z_i = g'^{-1}(x_i)(y_i - e(x_i)))$, $g'(x)$ is the derivation of $g(x)$. Obviously $z_i \neq 0$.

According to (60) of [8], we have the initial Gröbner basis in the form of $G_z = \{g_0 = 1, g_1 = z, \ldots, g_v = z^v, g_{v+1} = z^{v+1}b_1(x), g_{v+2} = z^{v+2}b_1^2(x)b_2(x), \ldots, g_L = z^L \prod_{i=1}^{L-v} b_i^{L-v-i+1}(x)\}$ after the re-encoding coordinate transformation. Obviously it's also a poset partially ordered by $\prec_+$, with one least element. Suppose the least element is $g_t = z^t \prod_{i=1}^{t-v} b_i^{t-v-i+1}(x), t \geq v$, then $g_t \prec_+ g_{t-1} \cdots \prec_+ g_v \cdots \prec_+ g_0$ and $g_t \prec_+ g_{t+1} \cdots \prec_+ g_L$. In the first chain, there may exist some adjacent elements' quotient that still can not be derivable at the transformed points. So the initial "update list" should contain all these elements to avoid adding them later, not only the least element $g_t$. Let $G = \{g_t, g_{t-1}, \ldots, g_v\}$, $p_j(x) = \prod_{i=1}^{j-v} b_i(x)$, then we have Algorithm 3 to interpolate the transformed points.

**Algorithm 3** Reduced-Complexity Interpolation Algorithm For Transformed Points

**Input:**
Interpolation points $(x_1, z_1)$, $(x_2, z_2), \ldots, (x_{n-k'}, z_{n-k'})$, $z_i \neq 0, i = 1, 2, \ldots, n - k'$; multiplicity $m_1, m_2, \ldots, m_{n-k'}$; maximum $z$-degree $L$

**Output:**
A non-zero polynomial $Q(x, z)$ in the module generated by $G_z$, satisfying the interpolation problem with minimal $(1, k - k')$-degree, $k - k' < 0$

```
1:  $G = \{g_t, g_{t-1}, \ldots, g_v\}$, $u \leftarrow t$, $w \leftarrow v$
2:  for each $i$ in $[1, n - k']$ do
3:    for $(r, s)$ from $(0, 0)$ to $(m_i - 1, 0)$ by $(m_i - 1, 1)$ lex order do
4:      for each $g_j$ in $G$ do
5:        $\Delta_j \leftarrow D_{r,s}(g_j(x_i, z_i))$
6:      end for
7:      $J \leftarrow \{j | \Delta_j \neq 0\}$
8:      if $J \neq \emptyset$ then
9:        $j^* \leftarrow \arg\min_{j \in J}(g_j)$
10:       $f \leftarrow g_{j^*}$, $\Delta \leftarrow \Delta_{j^*}$
11:     end if
12:     for all $j \in J$ do
13:       if $j \neq j^*$ then
14:         $g_j \leftarrow g_j - \frac{\Delta_j}{\Delta} f$
15:       else
16:         if $j^* = w$ and $j^* > 0$ then
17:           $g_{j^*-1} \leftarrow (z^{-1} - z_i^{-1}) f$
18:           $G \leftarrow G \cup \{g_{j^*-1}\}, w \leftarrow w - 1$
19:         end if
20:         if $j^* = u$ and $j^* < L$ then
21:           $g_{j^*+1} \leftarrow (p_{j^*+1}(x) z - p_{j^*+1}(x_i) z_i) f$
22:           $G \leftarrow G \cup \{g_{j^*+1}\}, u \leftarrow u + 1$
23:         end if
24:         $g_{j^*} \leftarrow (x - x_i) f$
25:       end if
26:     end for
27:   end for
28: end for
29: $Q(x, z) = \arg\min_{g \in G}(g)$
```

## V. A Reduced-Complexity Algorithm for Multivariate Polynomial Interpolation in Positive Weighted Degree

We will generalize Algorithm 1 for multivariate polynomial interpolation. In field $\mathbb{F}$, let $\mathbb{F}[x^{[0]}, x^{[1]}, x^{[2]}, \ldots, x^{[M]}]$ denote the ring of the $(M+1)$-variate polynomial. The weighted degree of every variate is denoted by $0 < k_0 \leq k_1 \leq k_2 \leq \ldots \leq k_M$. Their maximum degree limit are $+\infty, L_1, L_2, \ldots, L_M \in \{\mathbb{N} \cup \{+\infty\}\}$. Then $\mathbb{F}_{L_1, L_2, \ldots, L_M}[x^{[0]}, x^{[1]}, x^{[2]}, \ldots, x^{[M]}]$ is a $\mathbb{F}[x^{[0]}]$-module, with Gröbner basis $G = \{R | R = \prod_{i=1}^{M}(x^{[i]})^{d_i}, 0 \leq d_i \leq L_i\}$. $G$ is a complex poset partially ordered by divisibility. But we just need to know every element's maximum factor in $G$ except itself. The answer can tell us the timing when an element has to be computed. Let $\deg_{x^{[i]}}(f)$ denote the maximum $x^{[i]}$-degree of polynomial $f$. For term $R = \prod_{i=1}^{M}(x^{[i]})^{d_i}$, we denote its first variate $FV(R) = \arg\min_{d_i \neq 0}(i)$, and $FV(1) = M$. Then we have the following conclusions.

**Lemma 5.** *For term $R \neq 1$, the maximum factor except $R$ is $R/x^{[FV(R)]}$.*

**Lemma 6.** *Term $R$ is the maximum factor of $\{P' | P' = Rx^{[i]}, 0 < i \leq FV(R)\}$ except themselves.*

The two lemmas are obvious, so we omit the proof. We get the interpolation algorithm in Algorithm 4 by their correctness.

## VI. Numerical Results

It seems to be a complicated task to get the accurate computation reduction between the presented algorithms and Koetter's algorithm. So we just compare the complexity of Algorithm 1 and Koetter's algorithm by list-decoding RS(63,15) code with multiplicity $m = 4$. The maximum list size $L$ is 9. We compare their average complexity of finite field operation including add and multiplication in the process of interpolation.

From Figure 1 and Figure 2, we notice the presented algorithm almost reduces the computation by a constant number.

## VII. Conclusion

This poset based algorithm can be used to optimize most Koetter's interpolation algorithms[7], saving some computation. The accurate computation reduction is the main problem we have to solve in future.

**Algorithm 4** Reduced-Complexity Multivariate Interpolation Algorithm In Positive Weighted Degree

**Input:**

Interpolation points $(x_1^{[0]}, x_1^{[1]}, \dots, x_1^{[M]})$,$(x_2^{[0]}, x_2^{[1]}, \dots, x_2^{[M]})$, …, $(x_n^{[0]}, x_n^{[1]}, \dots, x_n^{[M]})$; multiplicity $m_1, m_2, \dots, m_n$; maximum degree limit $L_1, L_2, \dots, L_M \in \mathbb{Z} \cup \{+\infty\}$

**Output:**

$Q(x^{[0]}, x^{[1]}, \dots, x^{[M]}) \in \mathbb{F}_{L_1,L_2,\dots,L_M}[x^{[0]}, x^{[1]}, \dots, x^{[M]}]$ with minimal $(k_0, k_1, k_2, \dots, k_M)$-degree, interpolating $(x_1^{[0]}, x_1^{[1]}, \dots, x_1^{[M]})$, $(x_2^{[0]}, x_2^{[1]}, \dots, x_2^{[M]})$,…, $(x_n^{[0]}, x_n^{[1]}, \dots, x_n^{[M]})$ with multiplicity $m_1, m_2, \dots, m_n$.

```
1:  $G \leftarrow \{g_0 = 1\}$,$u \leftarrow 0$
2:  for each $i$ in $[1, n]$ do
3:    for each $(r_0, r_1, r_2, \dots, r_M)$ from $(0, 0, \dots, 0)$ to $(m_i, 0, 0, \dots, 0)$ by $(m_i - 1, 1, \dots, 1, 1)$ lex order do
4:      for each $j$ in $[0, |G| - 1]$ do
5:        $\Delta_j = D_{r_0,r_1,r_2,\dots,r_M}(g_j(x_i^{(0)}, x_i^{(1)}, \dots, x_i^{(M)}))$
6:      end for
7:      $J \leftarrow \{j | \Delta_j \neq 0\}$, $u \leftarrow u + 1$
8:      if $J \neq \emptyset$ then
9:        $j^* \leftarrow \arg\min_{j \in J}(g_j)$
10:       $f \leftarrow g_{j^*}$, $\Delta \leftarrow \Delta_{j^*}$
11:     end if
12:     for all $j \in J$ do
13:       if $j \neq j^*$ then
14:         $g_j \leftarrow g_j - \frac{\Delta_j}{\Delta} f$
15:       else
16:         for each $v$ in $[1, FV(LT(f))]$ do
17:           if $L_v = +\infty$ or $deg_{x^{[v]}}(f) < L_v$ then
18:             $\{G = G \cup \{(x^{[v]} - x_i^{[v]})f\}$
19:           end if
20:         end for
21:         $g_{j^*} = (x^{[0]} - x_i^{[0]})f$
22:       end if
23:     end for
24:   end for
25: end for
26: $Q(x^{[0]}, x^{[1]}, \dots, x^{[M]}) = \arg\min_{g \in G}(g)$
```

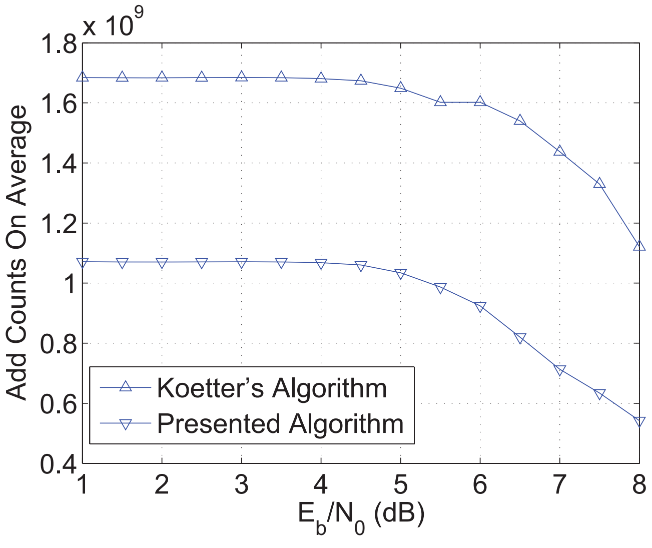

Figure 1. Average complexity of add

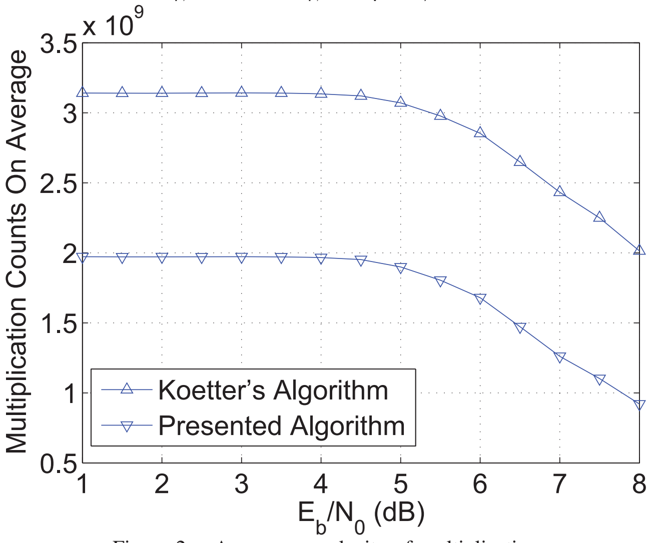

Figure 2. Average complexity of multiplication

## Acknowledgment

The authors are grateful to Prof. Xiao Ma and Dr. Shancheng Zhao from Sun Yat-sen University for useful discussions. They also have to thank Dr. Xiujie Huang and Xiaoxia Lu for financial support.

## References

[1] V. Guruswami and M. Sudan, "Improved decoding of Reed-Solomon and algebraic-geometric codes," in *Proceedings. 39th Annual Symposium on Foundations of Computer Science*, 1998, pp. 28–37.

[2] Y. Wu, "New List Decoding Algorithms for Reed-Solomon and BCH Codes," in *IEEE International Symposium on Information Theory*, June 2007, pp. 2806–2810.

[3] K. Lee and M. O'Sullivan, "An Interpolation Algorithm using Gröbner Bases for Soft-Decision Decoding of Reed-Solomon Codes," in *IEEE International Symposium on Information Theory*, July 2006, pp. 2032–2036.

[4] R. Koetter and A. Vardy, "Algebraic soft-decision decoding of reed-solomon codes," *Information Theory, IEEE Transactions on*, vol. 49, no. 11, pp. 2809 – 2825, Nov. 2003.

[5] S. Tang, L. Chen, and X. Ma, "Progressive list-enlarged algebraic soft decoding of Reed-Solomon codes," *Communications Letters, IEEE*, vol. 16, no. 6, pp. 901–904, June 2012.

[6] J. Ma and A. Vardy, "A Complexity Reducing Transformation for the Lee-O'Sullivan Interpolation Algorithm," in *IEEE International Symposium on Information Theory*, June 2007, pp. 1986–1990.

[7] R. Koetter, "Fast generalized minimum-distance decoding of algebraic-geometry and Reed-Solomon codes," *IEEE Transactions on Information Theory*, vol. 42, no. 3, pp. 721–737, May 1996.

[8] R. Koetter, J. Ma, and A. Vardy, "The Re-Encoding Transformation in Algebraic List-Decoding of Reed-Solomon Codes," *IEEE Transactions on Information Theory*, vol. 57, no. 2, pp. 633–647, Feb. 2011.

[9] B. Buchberger, *Gröbner-Bases: An Algorithmic Method in Polynomial Ideal Theory*. Reidel Publishing Company, Dodrecht - Boston - Lancaster, 1985.

[10] S. Sakata, "Finding a minimal set of linear recurring relations capable of generating a given finite two-dimensional array," *Journal of Symbolic Computation*, vol. 5, no. 3, pp. 321–337, 1988.