New Algebraic Soft Decision Decoding Algorithm for Reed-Solomon Code

Yuan Zhu
E-mail:41656955@qq.com

Abstract—In this paper, a new algebraic soft-decision decoding algorithm for Reed-Solomon code is presented. It is based on rational interpolation and the interpolation points are constructed by Berlekamp-Messay algorithm. Unlike the traditional Kötter-Vardy algorithm, new algorithm needs interpolation for two smaller multiplicity matrices, due to the corresponding factorization algorithm for re-constructing codewords.

Index Terms—Reed-Solomon code, rational interpolation, soft-decision decoding

I. INTRODUCTION

Reed-Solomon code, or RS code, is widely used in communication and digital storage. Its definition is based on the algebraic theory in finite field \([\mathbb{F}_q]\). So most of its decoding algorithms are based on related algebraic theory. Since Peterson’s invention[2] and other work [3], [4], the hard-decision bounded-decoding algorithms for RS code are widely used in practice. Modern RS code decoding algorithms, including list-decoding algorithm [5] and algebraic soft-decision decoding algorithm [6], enable better performance. These algorithms are based on polynomial interpolation with special multiplicity. A new list-decoding algorithm [7] for RS code based on rational interpolation has been introduced in 2008. It needs smaller interpolation multiplicity for the same list-decoding radius. In that paper, an open problem is raised: can we generalize the new algorithm for algebraic soft-decision decoding? In this paper, we give an algorithm to answer this question.

II. PREVIOUS WORK

The traditional error correction algorithm of \((n,k)\) RS code in \(GF(q)\) is to get an error locator polynomial for the error pattern. Suppose \(\alpha\) is a primitive element of \(GF(q)\) and \(n = q-1\). For a hard-decision vector of the receiving symbols, if it is different from the sending codeword at error locations set \(I\), then we have the error locator polynomial \(\Lambda(x) = \prod_{i \in I} (1 - \alpha^i x)\) for the codeword. \(\Lambda(x)\) or \(\Lambda\) for short, like other function in the following content) is also a connection polynomial for the syndromes sequence \((S_0, S_1, \ldots, S_{n-k})\) while the syndromes sequence is the first \(n-k\) part of the Fourier transformation of all probable error patterns.

The well known BM algorithm [3] can generate the error location polynomial \(\Lambda\), if the weight of error pattern is not beyond \([(n-k)/2]\). It always outputs two co-prime polynomial: one is \(\lambda\), the connection polynomial of syndromes sequence with minimal degree; the other is \(\delta\), the correction polynomial. What’s more, \(deg(\lambda) = n-k+1 = d\), \(d\) is the minimum distance of the RS code; \(\lambda\) and \(x\delta\) are co-prime. From Wu’s result [7, Lemma 3], any probable \(\Lambda\) can be expressed as \(\Lambda = \lambda u + x\delta v\, ,\ deg(\Lambda) = max(deg(\lambda u), deg(x\delta v))\), \(u\) and \(v\) are co-prime polynomials and \(u(0) = 1\). By the fact that the evaluation of \(\lambda u + x\delta v\) equals to zero at and only at the error locations, Wu’s algorithm uses a bivariate polynomial \(Q(x, y)\) with minimal \((1, deg(\lambda) + deg(\delta) - 1)\)-weighted degree to interpolate the \(n\) points \(\{(\alpha^{-i}, -\frac{\Lambda(\alpha^{-i})}{\alpha^{(t-1)}})|i = 1, 2, \ldots, n\}\) with special multiplicity (if \(\delta(\alpha^{-i}) = 0\), the \(i\)th interpolation point is \((\alpha^{-i}, \infty)\)). \(uy - v\) will be a factor of \(Q(x, y)\) if the weight of the error pattern is not beyond the list-decoding radius of Wu’s algorithm. As \(u(0) = 1\), when \(Q(x, y)\) has factor \(uy - v\), we can use Roth-Ruckenstein factorization algorithm [8] to get first \(deg(u) + deg(v)\) terms of Taylor series of \(v/u\) at zero and Padé approximation to get \(v/u\) [7].

Some relative algorithms generate the interpolation points not from BM algorithm, but extended Euclidean algorithm [9], [10]. In fact, they have the same performance for the same decoding radius as Wu’s algorithm in the worst case, either measured by the interpolation multiplicity or the list decoding size. It is usually ignored when the maximal \(y\)-degree of the interpolation result is greater than the multiplicity, the decoding radius is limited by the probable \([(n-k)/2 + 1]\) weight error pattern (from [7, Lemma 3]).

Something has to be mentioned before further discussion. First, as there may exist interpolation points at infinity, we can not use the traditional module based bivariate interpolation algorithm (including Kötter algorithm [11] and Lee-O’Sullivan algorithm [12]) to do the interpolation because the legal inner point of Wu’s algorithm is \((0, \infty)\), that is \(\Lambda(0) = \theta = 0\). Instead of the traditional interpolation, we use the Padé approximation to get list-decoding radius. In this paper, we can get Taylor series of \(v/u\) at zero and Padé approximation to get \(v/u\) [7].

The following result is organized as follows. A new interpolation points construction algorithm for list-decoding without infinity point is presented first. Then we generalize this algorithm for algebraic soft-decision decoding based on a \(n \times n\) multiplicity matrix. The codewords reconstruction and multiplicity assignment algorithms for it are also presented.

III. IMPROVED RATIONAL INTERPOLATION FOR WU’S ALGORITHM

We first construct new rational interpolation points without potential infinity. If \(deg(\lambda) \leq deg(x\delta)\), we denote \(a = \lambda\) and \(b = x\delta\); or else \(a = x\delta\) and \(b = \lambda\). Without loss of generality, suppose \(\Lambda = au + bv\, ,\) \(f(x) = b - \theta a, \theta \in GF(q)\) and we define \(f(x) = a\) if \(\theta = \infty\).

Lemma 1. There exist at least two different value \(\theta_1, \theta_2 \in (GF(q) \cup \{\infty\})\) to be \(\theta\), that \(f(x) = 0\) does not have any root in \(\{\alpha^{-1}, \alpha^{-2}, \ldots, \alpha^{-n}\}\).
\[ \theta = \frac{b(x)}{a(x)} \] Obviously, \( a/x \) can have at most \( n \) different evaluation results (including \( \infty \)) from \( x = \alpha^{-1}, \alpha^{-2}, \ldots, \alpha^{-n} \). As \( GF(q) \cup \{ \infty \} \) has two more elements than probable evaluation results, so there must exist at least two different values \( \theta_1, \theta_2 \in (GF(q) \cup \{ \infty \}) \) to be \( \theta \), that \( f(x) = 0 \) does not have any root in \( \{ \alpha^{-1}, \alpha^{-2}, \ldots, \alpha^{-n} \} \).

We can select one value from \( \theta_1 \) and \( \theta_2 \) as \( \theta \). For \( i \in I \), if \( \theta \neq \infty \), \( \Lambda = a(u + \theta v) + (b - \theta a)v \),

\[
\frac{v(\alpha^{-i})}{u(\alpha^{-i}) + \theta v(\alpha^{-i})} = -\frac{a(\alpha^{-i})}{b(\alpha^{-i}) - \theta a(\alpha^{-i})} \neq \infty
\]

If \( \theta = \infty \),

\[
\frac{u(\alpha^{-i})}{v(\alpha^{-i})} = -\frac{b(\alpha^{-i})}{a(\alpha^{-i})} \neq \infty
\]

Let's define \( g(x) \) and \( h(x) \):

\[
g(x) = \begin{cases} x^{n-1}, & \theta \neq \infty \\ x, & \theta = \infty \end{cases}
\]

\[
h(x) = \begin{cases} -\frac{a}{b - \theta a}, & \theta \neq \infty \\ -\frac{1}{\theta}, & \theta = \infty \end{cases}
\]

When a bivariate polynomial \( Q(x, y) \) interpolates the \( n \) points \( \{ (\alpha^{-1}, h(\alpha^{-1})), (\alpha^{-2}, h(\alpha^{-2})), \ldots, (\alpha^{-n}, h(\alpha^{-n})) \} \) at special multiplicity, then we can get the similar result like Wu's algorithm. We use \( \deg_{w_x, w_y}(Q) \) to denote the \( (w_x, w_y) \)-weighted degree of \( Q(x, y) \) so we can get its Taylor series at any point. But we can do some transformation to avoid the problem. If \( a = \lambda \) and \( \theta \neq \infty \), let \( Q(x, y) = (x - y)^{\delta E D}(Q) \), then we can do factorization from \( Q(x, y^{\delta E D}(Q)) \) to get factor \( u(x, y) \). If \( \theta = 0 \), we can directly factorize \( Q(x, y^{\delta E D}(Q)) \) to get factor \( u(x, y) \). If \( \theta = \infty \), then we can do factorization from \( Q(x, y^{\delta E D}(Q)) \) to get factor \( u(x, y) \). If \( \theta = \infty \), then we can directly factorize \( Q(x, y) \) to get factor \( u(x, y) \).

We need to pay attention that there are at least two choices of \( \theta \). For list-decoding in hard decision, we can choose any one. However, the other one is not useless. In the following soft-decision decoding algorithm, we need both of them to do interpolation twice.

IV. ALGEBRAIC SOFT DECISION DECODING ALGORITHM

A. Construction of Interpolation Points

Let's denote the Fourier transformation of the error pattern as polynomial \( E \). By the property of Fourier transformation and reverse, the value of error pattern at location \( \alpha^i \) is \( -E(\alpha^i) \). So \( \Lambda E \) has the factor \( \prod_{i=1}^{n} (1 - x^i) = 1 - x^n \). Then we can define the error evaluator polynomial \( \Omega \) that \( \Lambda E = \Omega(1 - x^n) \). Because the degree of \( E \) is less than \( n \), the degree of \( \Omega \) is less than \( \Lambda \). For any natural number \( \gamma \leq n \),

\[
(\Lambda E \mod x^\gamma) = \Omega \mod x^\gamma
\]

For list-decoding in hard decision, we can choose any one. However, the other one is not useless. In the following soft-decision decoding algorithm, we need both of them to do interpolation twice.

An alternative algorithm to solve the key-equation is called Berlekamp algorithm. It computes iteratively not only \( \Lambda \) but also \( \Omega \). While BM algorithm begins with \( \lambda = 1, \delta = 1 \), Berlekamp algorithm begins with \( \lambda = 1, \delta = 1, \omega = 0, \kappa = x^{-1} \). Berlekamp algorithm shares the same update rule of \( (\lambda, \delta) \) as in BM algorithm, and the update of \( (\omega, \kappa) \) also follows that rule. For example, if \( (\lambda, \delta) \) is updated as \( (\lambda + \Delta \omega, \delta + \lambda) \) in one iteration, then \( (\omega, \kappa) \) is updated as \( (\omega + \Delta \kappa, \kappa - \omega) \). We may notice \( \Omega = (1 + x^n + x^{2n} + \cdots) \), and \( \Delta \) is the second degree of Berlekamp algorithm. If \( a = \lambda \) and \( b = x \delta \), we denote \( c = 0 \) and \( d = x \kappa \); or else \( c = x \kappa \) and \( d = 0 \). By the fact that \( \Lambda \) and \( \Omega \) can be generated by the same transformation from \( (\lambda, \delta) \) and \( (\omega, \kappa) \) during the additional \( 2 \deg(\Lambda) - (n - k) \) iterations when \( 2 \deg(\Lambda) - (n - k) > 0 \), obviously we have the following conclusion.
Lemma 4. If \( \Lambda = au + bv \), then \( \Omega = cu + dv \).

For \( i \in I \), \( -E(\alpha^{-i}) = -\frac{\Omega(1-\alpha^{-i}^')}{\Lambda} \) by L'Hôpital's rule, or

\[
-E(\alpha^{-i}) = \frac{\Omega(\alpha^{-i})}{\alpha^{-i}\Lambda(\alpha^{-i})}
\]

\[
= -\frac{cu + dv}{x(uu + bv)} \mid_{x = \alpha^{-i}}
\]

We denote error value \( e_i = -E(\alpha^{-i}) \). When \( \theta \neq \infty \), as

\[
h(\alpha^{-i}) = \frac{-e_i(-\alpha^{-i})}{\alpha(-\alpha^{-i}) - \theta\alpha(-\alpha^{-i})} = \frac{u(-\alpha^{-i})}{x(-\alpha^{-i})} = g(\alpha^{-i}),
\]

\[
(-e_i\alpha^{-i})^{-1} = \frac{(a(u + \theta v) + (b - \theta a)v)'}{c(u + \theta v) + (d - \theta c)v} \mid_{x = \alpha^{-i}}
\]

\[
= \frac{\frac{b\theta}{\alpha} - \frac{b}{\alpha} - \frac{\theta}{\alpha}a}{\frac{d}{\alpha} + \frac{c}{\alpha}h} \mid_{x = \alpha^{-i}}
\]

When \( \theta = \infty \), \( h(\alpha^{-i}) = \frac{b\alpha(-\alpha^{-i})}{\alpha(-\alpha^{-i})} = \frac{u(-\alpha^{-i})}{x(-\alpha^{-i})} = g(\alpha^{-i}),
\]

\[
(-e_i\alpha^{-i})^{-1} = \frac{(au + bv)'}{c(u + \theta v) + (d - \theta c)v} \mid_{x = \alpha^{-i}}
\]

\[
= \frac{\frac{b\theta}{\alpha} - \frac{b}{\alpha} - \frac{\theta}{\alpha}a}{\frac{d}{\alpha} + \frac{c}{\alpha}h} \mid_{x = \alpha^{-i}}
\]

Let's define \( \phi(x) \):

\[
\phi(x) = \begin{cases}
\frac{-ad + bc}{\theta} & \text{if } \theta \neq \infty \\
\frac{ad - bc}{\theta} & \text{if } \theta = \infty
\end{cases}
\]

For \( i \in I \), we can evaluate the error value if we get \( g'(x) \):

\[
e_i = -\frac{\phi(\alpha^{-i})}{\alpha^{-i}(g'(\alpha^{-i}^{-1}) - h'(\alpha^{-i}^{-1}))}
\]

We use \( p(x, e) \) to construct the interpolation points for \( g'(x) \):

\[
p(x, e) = -(ex)^{-1}p + h', \quad e \neq 0, x \neq 0
\]

We can prove \( \phi(\alpha^{-i}) \) will never be zero for any \( i \). The numerator of \( \phi(x) \) is \( ad - bc \). If \( b(\alpha^{-i}) + \Delta a(\alpha^{-i}) \neq \frac{d(\alpha^{-i}) + \Delta c(\alpha^{-i})}{\alpha^{-i}} \), then

\[
\frac{b(\alpha^{-i}) + \Delta a(\alpha^{-i})}{\alpha^{-i}} - \frac{d(\alpha^{-i}) + \Delta c(\alpha^{-i})}{\alpha^{-i}} \neq \frac{e(\alpha^{-i})}{\alpha^{-i}}
\]

For \( a = 1, b = x, c = 0, d = 1, \frac{b(\alpha^{-i})}{\alpha^{-i}} \neq \frac{d(\alpha^{-i})}{\alpha^{-i}} \) for \( i = 1, 2, \ldots, n \). So by induction in Berlekamp algorithm, \( \phi(\alpha^{-i}) \) will never be zero for any \( i \). In another word, for different value of \( e \) and fixed value of \( x \), the evaluation of \( p(x, e) \) is different.

If we can have a multiplicity assignment function \( M(x, e) \), which maps from \( F^2 \) to \( Z^+ \cup \{0\} \) while \( F = GF(q^n) - \{0\} \), then we can do the soft-decision rational interpolation based on \( M(x, e) \).

**Theorem 5. For \( \theta \neq \infty \), \( Q(x, y) \) interpolates \((\alpha^{-i}, p(\alpha^{-i}, \alpha^{-j}))\) with multiplicity \( M(\alpha^{-i}, \alpha^{-j}) \) for every \( i, j \in \{1, 2, 3, \ldots, n\} \). If \( D = \deg_{1, \deg(a) - \deg(b) - 1}(Q) \) and \( D + 2(\deg(\Delta) - \deg(a))D_{Y(Q)} < \sum_{i \in I} M(\alpha^{-i}, e_i) \), then \( Q(x, g'(x)) = 0 \).

**Proof:** If \( \theta \neq \infty \), \( g'(x) = \frac{uv' - uv}{(uv')'} \), \( \deg(uv' - uv) \leq 2\deg(\Delta) - (\deg(a) + \deg(b)) - 1 \), \( \deg((u + \theta v)^2) \leq 2\deg(\Delta) - \deg(a) \); then the maximum degree of numerator of \( Q(x, g'(x)) \) is \( D + 2(\deg(\Delta) - \deg(a))D_{Y(Q)} \) like the bound in Theorem 2. For \( i \in I \), \( p(\alpha^{-i}, e_i) = g'(\alpha^{-i}) \), so the numerator of \( Q(x, g'(x)) \) passes \( x = \alpha^{-i} \) with multiplicity \( M(\alpha^{-i}, e_i) \). Once the degree of a polynomial is less than the sum of its zero point multiplicity, it must be the zero polynomial. So we prove the conclusion.

**Corollary 6. For \( \theta = \infty \), \( Q(x, y) \) interpolates \((\alpha^{-i}, p(\alpha^{-i}, \alpha^{-j}))\) with multiplicity \( M(\alpha^{-i}, \alpha^{-j}) \) for every \( i, j \in \{1, 2, 3, \ldots, n\} \). If \( D = \deg_{1, \deg(a) - \deg(b) - 1}(Q) \), then \( D + 2(\deg(\Delta) - \deg(b))D_{Y(Q)} < \sum_{i \in I} M(\alpha^{-i}, e_i) \), then \( Q(x, g'(x)) = 0 \).

During the interpolation to get \( Q(x, y) \), we have to solve

\[
C = \sum_{i=1}^{n} \sum_{j=1}^{n} M(\alpha^{-i}, \alpha^{-j})(M(\alpha^{-i}, \alpha^{-j}) + 1)
\]

(3) equations in total. Then we can have the following conclusion.

**Theorem 7. If**

\[
\frac{C + 1}{D_{Y + 1}} + D_{Y_{2}}((D_{Y + 1} - (2\deg(a) + d + 1)2) \leq \sum_{i \in I} M(\alpha^{-i}, e_i)
\]

(4)

and

\[
(\deg(b) + 1 - \deg(a))D_{Y + 1} \leq C + 1 \]

(5)

, then there exist a non-zero polynomial \( Q(x, y) \) with \( y \)-degree not greater than \( D_{Y} \), that it interpolates \((\alpha^{-i}, p(\alpha^{-i}, \alpha^{-j}))\) with multiplicity \( M(\alpha^{-i}, \alpha^{-j}) \) for \( i, j \in [1, n] \cap Z \) and \( Q(x, g'(x)) = 0 \).

**Proof:** If \( \theta \neq \infty \), suppose \( \deg_{1, \deg(b) - \deg(a) - 1}(Q) = D \) and \( D \geq 0 \), then \( Q(x, y) \) can have terms no more than \( (D_{Y + 1}) + (\deg(b) + 1 - \deg(a))D_{Y + 1}/2 \). So there exists \( D \geq 0 \) and

\[
(D_{Y + 1}) + (\deg(b) + 1 - \deg(a))D_{Y + 1}/2 \leq C + 1
\]

(5)

, then we complete the proof because \( (4) \) is stronger than \( (5) \) and \( \deg(b) + \deg(a) = d \).

If \( \theta = \infty \), then

\[
\frac{C + 1}{D_{Y + 1}} + D_{Y_{2}}((D_{Y + 1} - (2\deg(b) + d + 1)2) \leq \sum_{i \in I} M(\alpha^{-i}, e_i)
\]

(5)

because \( \deg(a) \leq \deg(b) \). So the conclusion above is true when \( \theta = \infty \) as well.
B. Improved Interpolation Result and Corresponding Codewords Reconstruction Algorithm in $GF(2^m)$

Usually, we only use RS code in $GF(2^m)$ in practice. In this field, the derivation of a polynomial must be a squared polynomial, and every element must have a unique square root. So we do not need to construct interpolation points for $g'(x)$ but its square root. We skip the proofs of the following conclusions because they are almost the same as the previous two, except some degree arguments change.

**Theorem 8.** In $GF(2^m)$, suppose $Q(x, y)$ interpolates $(\alpha_i^{-1}, \sqrt{P(\alpha_i^{-1}, \alpha_j^{-1})})$ with multiplicity $M(\alpha_i^{-1}, \alpha_j^{-1})$ for every $i, j \in \{1, 2, 3, \ldots, n\}$. If $\theta \neq \infty$, $D = \deg_{1, 0, 5}(\deg(a) - \deg(b) - 1)(Q)$ and

$$D + (\deg(A) - \deg(a))D_Y(Q) < \sum_{i \in I} M(\alpha_i^{-1}, e_i)$$

, then $Q(x, \sqrt[\alpha_i]{g'(x)}) = Q(x, \sqrt[\alpha_i]{\frac{u - \alpha_i}{u + \alpha_i}}) = 0$; or else if $\theta = \infty$, $D = \deg_{1, 0, 5}(\deg(b) - \deg(a) - 1)(Q)$ and

$$D + (\deg(A) - \deg(b))D_Y(Q) < \sum_{i \in I} M(\alpha_i^{-1}, e_i)$$

, then $Q(x, \sqrt[\alpha_i]{g'(x)}) = Q(x, \sqrt[\alpha_i]{\frac{u - \alpha_i}{u + \alpha_i}}) = 0$.

**Theorem 9.** In $GF(2^m)$, if

$$\frac{C + 1}{D_Y + 1} + D_Y(\deg(A) - (2\deg(a) + d + 1)/4) \leq \sum_{i \in I} M(\alpha_i^{-1}, e_i)$$

(6)

and

$$(\deg(b) + 1 - \deg(a))D_Y(D_Y + 1)/4 \leq C + 1$$

(7)

, then there exist polynomial $Q(x, y)$ with $y$-degree not greater than $D_Y$, that it interpolates $(\alpha_i^{-1}, \sqrt[\alpha_i]{P(\alpha_i^{-1}, \alpha_j^{-1})})$ with multiplicity $M(\alpha_i^{-1}, \alpha_j^{-1})$ for every $i, j \in \{1, 2, 3, \ldots, n\}$ and $Q(x, \sqrt[\alpha_i]{g'(x)}) = 0$.

We can set

$$D_Y = \left[\frac{2}{\sqrt{\deg(b) + 1 - \deg(a)} - 1}\right]$$

during the interpolation in the theorem. Larger $D_Y$ setting will make the interpolation result with minimal weighted degree has redundant $y$ factor ($D < 0$), which is a corollary from the negative weighted degree interpolation algorithm in [13].

Because the denominator of $\sqrt[\alpha_i]{g'(x)}$ may has factor $x$, we should not directly factorize $Q(x, y)$ to get $\sqrt[\alpha_i]{g'(x)}$. If $\theta \neq \infty$ and we are concerned to $\Lambda$ that $\deg(A) \leq \rho$ and $\rho \geq b$, the worst case is $u + \theta v = x^{\rho - \deg(a)}$. So we should factorize the polynomial $x^{(\rho - \deg(a))}D_Y Q(x, x^{(\rho - \deg(a))}y)$ instead. When we use Roth-Ruckenstein factorization algorithm to get the terms of $x^{(\rho - \deg(a))}D_Y \sqrt[\alpha_i]{g'(x)}$, suppose the degree of the first non-zero term is $\rho' - \deg(a)$, then we need to get next $\rho - \deg(a) + [(2\rho - \deg(a) - \deg(b) - 1)/2] - [\rho - \rho'] - 1$ terms at most for the demand of Padé approximation, because the denominator of $\sqrt[\alpha_i]{g'(x)}$ must has factor $x^{\rho - \rho'}$ if $\rho > \rho'$.

Once we get $g'(x)$, we can not recover $g(x)$ from $g'(x)$ directly in finite field. But we can use two “curves” to get the “intersection point”. Suppose $\theta_1 \neq \infty$. For the RS code in $GF(2^m)$, we can construct the interpolation points for both $\theta = \theta_1$ and $\theta = \theta_2$ to do interpolation twice and get the two interpolation result $Q_1(x, y), Q_2(x, y)$. As (8) is not related to the value of $\theta$, when the precondition is satisfied, we can get both $\sqrt[\alpha_i]{g_1} = \sqrt[\alpha_i]{\frac{u - \alpha_i}{u + \alpha_i}}$ from factorization of $Q_1(x, y)$ and $g_2 = \sqrt[\alpha_i]{\frac{u - \alpha_i}{u + \alpha_i}}$ if $\theta_2 \neq \infty$ or else $g_2 = \sqrt[\alpha_i]{\frac{u - \alpha_i}{u + \alpha_i}}$ from $Q_2(x, y)$. If $\sqrt[\alpha_i]{g_1}/\sqrt[\alpha_i]{g_2}$ is known, we can get $\frac{u}{u + \alpha_i}$. We will introduce two factorization algorithms to get $\sqrt[\alpha_i]{g_1}/\sqrt[\alpha_i]{g_2}$.

The first one is algebraic. $Q_1(x, y)$ and $Q_2(x, y)$ can be viewed as two univariate polynomials in polynomial ring for variable $y$, with coefficients of elements in univariate rational function field for variable $x$. $Q_1(x, z)$ and $Q_2(x, y)$ have the common factor $y - \sqrt[\alpha_i]{g_2}$ in the ring if $z = \sqrt[\alpha_i]{g_1}/\sqrt[\alpha_i]{g_2}$. We can use Euclidean algorithm from $Q_1(x, z)$ and $Q_2(x, y)$ to get $R(x, z)$ after elimination variable $y (R(z, x)$ is the resultant of $Q_1(x, z)$ and $Q_2(x, y)$ for variable $y$). By the property of Euclidean domain, $R(x, (\sqrt[\alpha_i]{g_1}/\sqrt[\alpha_i]{g_2}))$ also has factor $y - \sqrt[\alpha_i]{g_2}$.

In another word, $R(x, \sqrt[\alpha_i]{g_1}/\sqrt[\alpha_i]{g_2}) = 0$ (this is the property of resultant). So we can factorize $R(z, x)$ to get the root $\sqrt[\alpha_i]{g_1}/\sqrt[\alpha_i]{g_2}$. The $z$-degree of $R(z, x)$ is no more than $D_2^2/2$.

The other algorithm needs to do rational factorization twice. We can factorize both $Q_1(x, y)$ and $Q_2(x, y)$ for variable $y$. Then $\sqrt[\alpha_i]{g_1}/\sqrt[\alpha_i]{g_2}$ must be the division result that one root of $Q_2(x, y)$ divided by one root of $Q_1(x, y)$. So we can test all the pair combination of their roots no more than $D_2^2$ times. Though this algorithm may output the same result as the previous after we list all the probable division result, the $y$-degree of $Q_1(x, y)$ and $Q_2(x, y)$ is not greater than $D_Y$ and the complexity of factorization reduces significantly.

Once we get $\sqrt[\alpha_i]{g_1}/\sqrt[\alpha_i]{g_2}$, we can get $G(x) = (\theta_2 - \theta_1)^{-1}(\sqrt[\alpha_i]{g_1} - 1)$ if $\theta_2 \neq \infty$ or else $G(x) = \sqrt[\alpha_i]{g_1}$. Then we need to check whether $G(\alpha_i^{-1}) = h(\alpha_i^{-1})$ for $i = 1, 2, \ldots, n$ to construct the error location set $I$. If the evaluation of $\sqrt[\alpha_i]{g_1}$ is $q_0$ type, we should remove the redundant factor in the numerator and denominator. After we get $I$, if $|I| \leq n - k$, we can interpolate any correct $k$ positions to get the codeword polynomial; else for $i \in I$, we can use (2) to evaluate the error value $e_i$, then check whether it is a valid error pattern.

For the RS code in a field with character beyond two, however, we can only get $g_1'/g_2'$ after factorization of the two interpolation result. To detect the error location, we have to factorize the denominator and numerator of $g_1'/g_2'$ to get $\sqrt[\alpha_i]{g_1}/\sqrt[\alpha_i]{g_2}$. This is because we can not compare the evaluation of $g_1'/g_2' = (\frac{u - \alpha_i}{u + \alpha_i})^2$ and $(\frac{u - \alpha_i}{u + \alpha_i})^2$ if $\theta_2 \neq \infty$, because different values in this field may have same square result.

V. Multiplicity Assignment in $GF(2^m)$

In Kötter-Vardy algorithm, a multiplicity assignment algorithm [6, Algorithm A] is given to ensure the performance of
the decoding algorithm. We will prove that multiplicity assignment algorithm can also be used in our decoding algorithm. For practical reason, we just discuss it in $\text{GF}(2^m)$.

As $\deg(\Lambda) = |I|$, from [6] we have

$$\frac{C + 1}{D_Y + 1} - \sum_{i \in I} (M(\alpha^{-i}, e_i) - D_Y))D_Y^{-1} \leq \frac{2 \deg(a) + d + 1}{4}$$

(8)

Suppose the memoryless channel outputs symbols $Z$ and the receiver receives a vector $z$ in $Z^n$. We denote hard-decision of $z$ is $\hat{z}$. Error pattern vector $e$ is the subtraction of $\hat{z}$ and the sending codeword. We use a $n \times n$ matrix $II$ to store the posteriori probability for the $n$ non-zero error values at $n$ positions, $\Pi(\alpha^{-i}, e_i) = \Pr(\hat{z}_i - e_i | z_i)$. The multiplicity assignment algorithm is to find a multiplicity assignment function $M(x, e)$ according to matrix $II$. We denote the output of function $M(x, e)$ by a $n \times n$ integer matrix $M$ for $e \neq 0$ and we define $M(x, 0) = D_Y$. Then we can define the random variable for the left side of (8)

$$W(M) = \left(\frac{C + 1}{D_Y + 1} - \sum_{i=1}^{n} (M(\alpha^{-i}, e_i) - D_Y))D_Y^{-1}\right)$$

(9)

We can use Markov inequality to maximize the lower bound of probability of (8). In another word, we want to minimize the expectation of $W(M)$ like Kötter-Vardy’s method under fixed $C$ and $D_Y$, Using the theory model in [16], [9], we research all the coset codes of the RS code on average instead to simplify the model. Then error values $e_i$ $(i = 1, 2, \ldots, n)$ can be viewed as independent and identically distributed random variables, with prior probability in uniform distribution in finite field. By the linearity of expectation, the expectation of $W(M)$, $E(W(M))$, is

$$E(W(M)) = \left(\frac{C + 1}{D_Y + 1} - \sum_{i=1}^{n} \sum_{j=1}^{n} \Pi(\alpha^{-i}, \alpha^{-j}) (M(\alpha^{-i}, \alpha^{-j}) - D_Y))D_Y^{-1}\right)$$

So we just need to get $M$ to maximize the inner product of $M$ and $II$, $(M, II)$. In another word, we can reuse the multiplicity assignment algorithm[6] Algorithm A] to get the best $M$ and minimize the expectation of $W(M)$ from II.

Using the Kötter-Vardy’s multiplicity assignme algorithm, when $C \to \infty$, $M \to sII$ if we enable the elements of $M$ has real number(except for $M(x, 0)$) and $s$ is a number related to $C$. Then $D_Y \to 2\sqrt{\frac{C}{d+1-2\deg(a)}}$, $s \to \sqrt{\frac{2C}{\Pi, II}}$ because of (8). So we can conclude from (8) asymptotically that our algebraic soft-decision decoding algorithm outputs a list that contains an error vector $e$ if

$$\deg(\Lambda) - \deg(a) + o(1) \leq \sqrt{\frac{d + 1 - 2\deg(a)}{2}} \sum_{i \in I} \Pi(\alpha^{-i}, e_i)$$

(9)

It’s not easy to give a fair compare between [9] and [6] Theorem 12], but obviously when Kötter-Vardy’s algorithm does not output any codeword, [9] shows our soft decision decoding algorithm may have output. So at least, this algorithm can be considered as an alternative method for RS decoding when other decoding algorithms fail.

VI. CONCLUSIONS

This paper briefly introduces a new algebraic soft-decision decoding algorithm for RS code based on rational interpolation and factorization, answering the open problem in [2]. But there are several important work left. We need more detailed performance analyze for the presented multiplicity assignment algorithm according to Markov inequality. We need to find some better multiplicity assignment algorithms, such as Chernoff bound type algorithm like [10] and types method like [17] for our decoding algorithm. The factorization needs better algorithm to reduce the complexity. An open question is left: can we find the re-encoding algorithm like [18] to reduce the complexity of rational interpolation?

REFERENCES

[1] I. S. Reed and G. Solomon, “Polynomial codes over certain finite fields,” Journal of the Society for Industrial & Applied Mathematics, vol. 8, no. 2, pp. 300–304, 1960.
[2] W. Peterson, “Encoding and error-correction procedures for the bose-chaudhuri codes,” Information Theory, IRE Transactions on, vol. 6, no. 4, pp. 459–470, September 1960.
[3] J. Massey, “Shift-register synthesis and bch decoding,” Information Theory, IEEE Transactions on, vol. 15, no. 1, pp. 122–127, Jan 1969.
[4] Y. Sugiyama, M. Kasahara, S. Hirasawa, and T. Namekawa, “An erasures-and-errors decoding algorithm for goppa codes (corresp.),” Information Theory, IEEE Transactions on, vol. 22, no. 2, pp. 238–241, Mar 1976.
[5] V. Guruswami and M. Sudan, “Improved decoding of Reed-solomon and algebraic-geometry codes,” Information Theory, IEEE Transactions on, vol. 45, no. 6, pp. 1757–1767, Sep 1999.
[6] R. Kötter 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.
[7] Y. Wu, “New list decoding algorithms for Reed-solomon and BCH codes,” Information Theory, IEEE Transactions on, vol. 54, no. 8, pp. 3611–3630, Aug 2008.
[8] R. Roth and G. Ruckenstein, “Efficient decoding of Reed-solomon codes beyond half the minimum distance,” in Information Theory, 1998. Proceedings. 1998 IEEE International Symposium on, Aug 1998, pp. 56–64.
[9] M. Ali and M. Kuijper, “A parametric approach to list decoding of Reed-solomon codes using interpolation,” Information Theory, IEEE Transactions on, vol. 57, no. 10, pp. 6718–6728, Oct 2011.
[10] P. Trifonov and M. ho Lee, “Efficient interpolation in the wu list decoding algorithm,” Information Theory, IEEE Transactions on, vol. 58, no. 9, pp. 5963–5971, Sept 2012.
[11] R. Kötter, “Fast generalized minimum-distance decoding of algebraic-geometry and Reed-solomon codes,” Information Theory, IEEE Transactions on, vol. 42, no. 3, pp. 721–737, May 1996.
[12] K. Lee and O’Sullivan, “List decoding of reed–solomon codes from a gröbner basis perspective,” Journal of Symbolic Computation, vol. 43, no. 9, pp. 645–658, 2008.
[13] E. R. Berlekamp, Algebraic coding theory, vol. 111.
[14] R. E. Blahut, Algebraic codes on lines, planes, and curves. Cambridge University Press Cambridge, 2008.
[15] Y. Zha and S. Tang, “A reduced-complexity algorithm for polynomial interpolation,” in Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on, July 2013, pp. 316–320.
[16] M. El-Khany and R. J. McEliece, “Interpolation multiplicity assignment algorithms for algebraic soft-decision decoding of reed-solomon codes,” Algebraic Coding Theory and Information Theory, pp. 99–120, 2005.
[17] H. Das and A. Vardy, “Multiplicity assignments for algebraic soft-decoding of Reed-solomon codes using the method of types,” in Information Theory, 2009. ISIT 2009. IEEE International Symposium on, June 2009, pp. 1248–1252.
[18] R. Koetter, J. Ma, and A. Vardy, “The re-encoding transformation in algebraic list-decoding of Reed-solomon codes,” Information Theory, IEEE Transactions on, vol. 57, no. 2, pp. 633–647, Feb 2011.