Algebraic Chase Decoding of Reed-Solomon Codes Using Module Minimisation

Li Chen †, Martin Bossert ‡

† School of Electronics and Information Technology, Sun Yat-sen University, Guangzhou, China
‡ Institute of Communications Engineering, Ulm University, Ulm, Germany
Email: chenli55@mail.sysu.edu.cn, martin.bossert@uni-ulm.de

***Abstract*—This paper proposes a low-complexity high performance soft-in hard-out decoding algorithm for Reed-Solomon (RS) codes. The Guruswami-Sudan (GS) algebraic list decoding algorithm can correct errors beyond half the distance bound by performing a curve-fitting decoding process. However, its extra error-correction capability is exchanged with a high computational cost which is dominated by the interpolation. In this paper, an algebraic Chase decoding (ACD) approach that utilises the module minimisation (MM) technique is introduced, namely the ACD-MM algorithm. The MM technique solves the interpolation problem with less computational cost than the conventional Koetter's interpolation. The proposal exploits the soft received information to formulate the interpolation test-vectors. The re-encoding transform is further employed to reduce the size of the entries of the module, benefiting a simpler MM process. Our analyses show that the ACD-MM algorithm can outperform the legacy Koetter-Vardy (KV) soft-decision list decoding algorithm with a much lower complexity. Moreover, the proposal allows parallel execution of each Chase decoding trial, resulting in a low decoding latency for practical interest.**

***Index Terms*—Algebraic decoding, Chase decoding, interpolation, module minimisation, Reed-Solomon codes**

I. Introduction

Reed-Solomon (RS) codes are widely employed in wireless communications and data storage systems. In industry, the conventional Berlekamp-Massey (BM) algorithm [1] is used. It has an efficient running time but with error-correction capability limited by half of the minimum Hamming distance of the code. In late 90s, Guruswami and Sudan introduced the algebraic list decoding, namely the GS algorithm [2], which can correct errors beyond the half distance bound. It is a curve-fitting approach that includes interpolation and factorisation, while the former dominates the complexity. Soft-decision list decoding was later introduced by Koetter and Vardy, namely the KV algorithm [3], offering a significant soft decoding gain.

However, the algebraic list decoding algorithm's high complexity prevents its more immediate applications. This is mainly due to the interpolation that finds the minimal interpolated polynomial. It is often realised using Koetter's algorithm [4] which is an iterative polynomial construction approach. The main complexity reduction methods include the re-encoding transform [5], Wu's algorithm [6] and the low-complexity Chase (LCC) algorithm [7]. Meanwhile, the average decoding complexity issue has been addressed by [8] and [9] that proposes a progressive list decoding mechanism.

The interpolation problem can also be solved from the perspective of Gröbner basis of module [10]. In contrast to Koetter's interpolation that grows the entries of the Gröbner basis in a point-by-point fashion, one can define a module with polynomials that interpolate all the prescribed points. The Mulders-Storjohann (MS) algorithm [11] can be further deployed to reduce the module into the weak Popov form, resulting in the desired Gröbner basis from which the minimal interpolated polynomial can be found. Such an interpolation technique is called module minimisation (MM) and it is less complex than Koetter's interpolation. The subsequent MM improvements for decoding RS codes include the divide-and-conquer approach [12] and its further modified parameterisation [13]. Recently, a MM based multi-trial GS decoding approach was developed in [14]. On the other aspect, Chase decoding of RS codes using the extended Euclidean algorithm has recently been reported in [15].

Inspired by the MM interpolation technique, this paper proposes a new algebraic Chase decoding (ACD) algorithm for RS codes, namely the ACD-MM algorithm. Soft received information is exploited to formulate the interpolation test-vectors. Such a formulation paves the way of performing the re-encoding transform which results in a simpler MM process by reducing the size of the module entries. Our simulation results and analyses show that the proposal can outperform most of the decoding algorithms for RS codes with a complexity that is orders of magnitude less than the KV algorithm. In addition, the ACD-MM algorithm allows parallel execution for all the Chase decoding trials, granting a low decoding latency which is of practical interest.

II. RS Codes and List Decoding

This section reviews the RS encoding and its GS list decoding utilising MM.

A. RS Codes

Let $\mathbb{F}_q = \{0, 1, 2, \ldots, q-1\}$ denote the finite field of size q, and $\mathbb{F}_q[x]$ and $\mathbb{F}_q[x, y]$ denote the univariate and bivariate polynomial rings defined over $\mathbb{F}_q$, respectively. For an (n, k) RS code, where $n = q-1$ and k are the length and dimension of the code, respectively, the message polynomial $f(x) \in \mathbb{F}_q[x]$ can be written as

$$f(x) = f_0 + f_1 x + \cdots + f_{k-1} x^{k-1}, \tag{1}$$

where $f_0, f_1, \ldots, f_{k-1}$ are the message symbols. The codeword $\underline{c} = (c_0, c_1, \ldots, c_{n-1}) \in \mathbb{F}_q^n$ can be generated by

$$\underline{c} = (f(\alpha_0), f(\alpha_1), \ldots, f(\alpha_{n-1})), \tag{2}$$

where $\alpha_0, \alpha_1, \ldots, \alpha_{n-1}$ are n distinct nonzero elements of $\mathbb{F}_q$, and they are called the code locators.

B. List Decoding Using MM

Let $\underline{\omega} = (\omega_0, \omega_1, \ldots, \omega_{n-1}) \in \mathbb{F}_q^n$ denote the hard-decision received word. To perform interpolation, the following n points can be formed as

$$(\alpha_0, \omega_0), (\alpha_1, \omega_1), \ldots, (\alpha_{n-1}, \omega_{n-1}). \tag{3}$$

Let us define the Hamming distance between codeword $\underline{c}$ and a received word $\underline{\omega}$ as

$$d_{\mathrm{H}}(\underline{c}, \underline{\omega}) = |\{j \mid c_j \neq \omega_j, \forall j\}|. \tag{4}$$

Furthermore, given a polynomial $Q(x, y) = \sum_{a,b} Q_{ab} x^a y^b \in \mathbb{F}_q[x, y]$, its (μ, ν)-weighted degree is defined as

$$\deg_{\mu,\nu} Q(x, y) = \max\{\mu a + \nu b \mid Q_{ab} \neq 0\}. \tag{5}$$

Armed with the above prerequisites, the following GS list decoding theorem can be introduced.

Theorem 1: For an (n, k) RS code, let $Q \in \mathbb{F}_q[x, y]$ denote the interpolated polynomial that passes through the n points of (3) with a multiplicity of m, and it has a maximal y-degree of l[1]. If $m(n - d_{\mathrm{H}}(\underline{c}, \underline{\omega})) > \deg_{1,k-1} Q$, then $Q(x, f(x)) = 0$ [2].

Interpolation is to find such a polynomial Q with the minimal $(1,\ k-1)$-weighted degree. Factorisation can be further performed to find its y-roots [16]. Now, we start to describe the MM technique for solving the interpolation problem.

Definition I: Let $\underline{\xi} = (\xi_0(x), \xi_1(x), \ldots, \xi_t(x))$ denote a vector over $\mathbb{F}_q[x]$, we define the degree of $\underline{\xi}$ as

$$\deg \underline{\xi} = \max\{\deg \xi_i(x), i = 0, 1, \ldots, t\}. \tag{6}$$

The leading position (LP) of vector $\underline{\xi}$ is defined as

$$\mathrm{LP}(\underline{\xi}) = \max\{i \mid \deg \xi_i(x) = \deg \underline{\xi}\}, \tag{7}$$

and hence the leading term (LT) of vector $\underline{\xi}$ is

$$\mathrm{LT}(\underline{\xi}) = \xi_{\mathrm{LP}(\underline{\xi})}(x). \tag{8}$$

Since the entries of $\underline{\xi}$ can be elaborated as $\xi_i(x) = \xi_i^{(0)} + \xi_i^{(1)} x + \cdots + \xi_i^{(\deg \xi_i(x))} x^{\deg \xi_i(x)}$, the leading coefficient (LC) of $\xi_i(x)$ can be further defined as

$$\mathrm{LC}(\xi_i(x)) = \xi_i^{(\deg \xi_i(x))}. \tag{9}$$

For GS decoding with an interpolation multiplicity of m and it delivers at most l message candidates, a module $\mathcal{M}_{m,l}$ which can be seen as a $(l+1) \times (l+1)$ matrix over $\mathbb{F}_q[x]$ is required. The module is defined as follows.

Definition II: A module $\mathcal{M}_{m,l}$ is the space of all bivariate polynomials over $\mathbb{F}_q[x, y]$ that interpolate the n points with a multiplicity of m and have the maximal y-degree of l.

[1]Parameterisation of m and l can be seen at [2, 14] and it holds that $m \leq l$.

In order to constitute module $\mathcal{M}_{m,l}$, the following two polynomials are needed.

$$G(x) = \prod_{j=0}^{n-1} (x - \alpha_j), \tag{10}$$

$$R(x) = \sum_{j=0}^{n-1} \omega_j L_j(x), \tag{11}$$

where $L_j(x) = \prod_{i=0, i\neq j}^{n-1} \frac{x-\alpha_i}{\alpha_j - \alpha_i}$ is a Lagrange basis polynomial. Consequently, the module $\mathcal{M}_{m,l}$ can be formed by the following $l+1$ generators.

$$P_t(x, y) = G(x)^{m-t}(y - R(x))^t, \text{if } 0 \leq t \leq m, \tag{12}$$

$$P_t(x, y) = y^{t-m}(y - R(x))^m, \text{if } m < t \leq l. \tag{13}$$

By realising $R(\alpha_j) = \omega_j, \forall\ j$, it can be seen that polynomial $P_t(x, y)$ interpolates the n points of (3) with a multiplicity of m and has y-degree of t $(t \leq l)$. Since $P_t(x, y)$ can also be written as $P_t(x, y) = \sum_{\tau \leq t} P_t^{(\tau)}(x) y^\tau$, where $P_t^{(\tau)}(x) \in \mathbb{F}_q[x]$, a module $\mathcal{M}_{m,l}$ can be constructed by fulfilling its entry of row-t column-τ with $P_t^{(\tau)}(x)$. For now, one can see module $\mathcal{M}_{m,l}$ is a square matrix over $\mathbb{F}_q[x]$.

Definition III: Let $\mathcal{D}_{\beta,l} = \mathrm{diag}(1, x^\beta, \ldots, x^{l\beta})$ and $\ddot{\mathcal{D}}_{\beta,l} = \mathrm{diag}(x^{l\beta}, x^{(l-1)\beta}, \ldots, 1)$ denote the diagonal matrices of size $(l+1) \times (l+1)$, where β is an integer. One can have two types of mapping for module $\mathcal{M}_{m,l}$ following

$$\mathcal{A}_{m,l} = \mathcal{M}_{m,l} \cdot \mathcal{D}_{\beta,l}, \tag{14}$$

$$\mathcal{A}_{m,l} = \mathcal{M}_{m,l} \cdot \ddot{\mathcal{D}}_{\beta,l}. \tag{15}$$

Note that $\mathcal{A}_{m,l}$ is no longer a module, but a square matrix over $\mathbb{F}_q[x]$. Inversely, the demapping of $\mathcal{A}_{m,l}$ can be

$$\mathcal{M}_{m,l} = \mathcal{A}_{m,l} \cdot \mathcal{D}_{-\beta,l}, \tag{16}$$

$$\mathcal{M}_{m,l} = \mathcal{A}_{m,l} \cdot \ddot{\mathcal{D}}_{-\beta,l}. \tag{17}$$

With the demapping of $\mathcal{A}_{m,l}$, a module is restored.

Definition IV: A matrix over $\mathbb{F}_q[x]$ is in *weak Popov form* if the leading position of each row is different [11].

For list decoding utilising MM, after the module $\mathcal{M}_{m,l}$ is formed by the generators of (12)(13), the mapping of

$$\mathcal{A}_{m,l} = \mathcal{M}_{m,l} \cdot \mathcal{D}_{k-1,l} \tag{18}$$

is performed. As a result, for a row $\underline{\xi}_t$ in $\mathcal{A}_{m,l}$, we have $\deg \underline{\xi}_t = \deg_{1,k-1} P_t(x, y)$. Afterwards, the following MS row reduction algorithm [11] is applied on $\mathcal{A}_{m,l}$, yielding $\mathcal{A}_{m,l}$ being in the weak Popov form and it is denoted as $\mathcal{A}'_{m,l}$. By further performing the demapping of

$$\mathcal{M}'_{m,l} = \mathcal{A}'_{m,l} \cdot \mathcal{D}_{-(k-1),l}, \tag{19}$$

the minimal interpolated polynomial Q can be taken from a row of $\mathcal{M}'_{m,l}$. This row corresponds to the row in $\mathcal{A}'_{m,l}$ with the minimal degree.

Algorithm 1 Mulders-Storjohann Algorithm

Input: $\mathcal{A}_{m,l}$;
Output: $\mathcal{A}'_{m,l}$;
1: While $\mathcal{A}_{m,l}$ is not in weak Popov form **do**
2: Find two rows $\underline{\xi}_a$ and $\underline{\xi}_b$ in $\mathcal{A}_{m,l}$ such that $\deg \underline{\xi}_a \leq \deg \underline{\xi}_b$ and $\mathrm{LP}(\underline{\xi}_a) = \mathrm{LP}(\underline{\xi}_b)$;
3: Perform $\underline{\xi}_b \leftarrow \underline{\xi}_b - \frac{\mathrm{LC}(\xi_{\mathrm{LP}(\underline{\xi}_b)}(x))}{\mathrm{LC}(\xi_{\mathrm{LP}(\underline{\xi}_a)}(x))} x^{\deg \underline{\xi}_b - \deg \underline{\xi}_a} \cdot \underline{\xi}_a$;
4: End while

III. The ACD-MM Algorithm

This section presents the ACD-MM algorithm. We start with formulation of the interpolation test-vectors.

A. Test-Vector Formulation

Let $\Pi \in \mathbb{R}^{q \times n}$ denote the reliability matrix observed from the channel. By assuming its rows are indexed by elements of $\mathbb{F}_q$, its entry π_{ij} is the *a posteriori* probability defined as $\pi_{ij} = \Pr[c_j = i]$, for $i = 0, 1, \ldots, q-1, j = 0, 1, \ldots, n-1$. By identifying the largest and the second largest entries of column j as $\pi_j^{\mathrm{I}} = \max_{i \in \mathbb{F}_q}\{\pi_{ij}\}$ and $\pi_j^{\mathrm{II}} = \max_{i \in \mathbb{F}_q}\{\pi_{ij} \mid \pi_{ij} \neq \pi_j^{\mathrm{I}}\}$, respectively, we can define

$$r_j^{\mathrm{I}} = i|_{\pi_{ij} = \pi_j^{\mathrm{I}}},\ r_j^{\mathrm{II}} = i|_{\pi_{ij} = \pi_j^{\mathrm{II}}} \tag{20}$$

as the most likely and the second most likely decisions for symbol c_j. Define the symbol wise reliability metric as [7]

$$\gamma_j = \frac{\pi_j^{\mathrm{II}}}{\pi_j^{\mathrm{I}}}, \tag{21}$$

where $\gamma_j \in (0, 1)$. With $\gamma_j \to 0$, the decision on c_j is more reliable, and vise versa. By sorting the n reliability metrics in an ascending order, we can have a refreshed symbol index sequence $j_0, j_1, \ldots, j_{n-1}$. It indicates $\gamma_{j_0} < \gamma_{j_1} < \cdots < \gamma_{j_{n-1}}$. Therefore, we can define $\Theta = \{j_0, j_1, \ldots, j_{k-1}\}$ as the indices for the reliable symbols and $\overline{\Theta} = \{j_k, j_{k+1}, \ldots, j_{n-1}\}$ as the indices for the unreliable ones. Choose η ($\eta \leq n - k$) unreliable symbols from $\overline{\Theta}$, and they can either be realised as r_j^{I} or r_j^{II}, we can formulate 2^η interpolation test-vectors which can be generally written as

$$\underline{r}_u = (r_{j_0}^{(u)}, r_{j_1}^{(u)}, \ldots, r_{j_{k-1}}^{(u)}, r_{j_k}^{(u)}, \ldots, r_{j_{n-1}}^{(u)}), \tag{22}$$

where $u = 1, 2, \ldots, 2^\eta$, among which $r_j^{(u)} = r_j^{\mathrm{I}}$ for $j = j_0, j_1, \ldots, j_{n-\eta-1}$ and $r_j^{(u)} = r_j^{\mathrm{I}}$ or r_j^{II} for $j = j_{n-\eta}, j_{n-\eta+1}, \ldots, j_{n-1}$.

B. Re-encoding Transform

The MM interpolation can be simplified by reducing the size (degree) of the module entries. This will make the row reduction in Algorithm 1 simpler with less finite field operations. The re-encoding transform serves this motivation. After the test-vector formulation, points $(\alpha_{j_0}, r_{j_0}^{\mathrm{I}}), (\alpha_{j_1}, r_{j_1}^{\mathrm{I}}), \ldots, (\alpha_{j_{k-1}}, r_{j_{k-1}}^{\mathrm{I}})$ are chosen for re-encoding. The re-encoding polynomial is defined as

$$H(x) = \sum_{j \in \Theta} r_j^{\mathrm{I}} h_j(x), \tag{23}$$

where $h_j(x) = \prod_{i \in \Theta, i \neq j} \frac{x - \alpha_i}{\alpha_j - \alpha_i}$ is again a Lagrange basis polynomial. All test-vectors $\underline{r}_u$ can be transformed by

$$\underline{r}_u \mapsto \underline{z}_u : z_j^{(u)} = r_j^{(u)} - H(\alpha_j), \forall j. \tag{24}$$

Since $H(\alpha_j) = r_j^{\mathrm{I}}, \forall\ j \in \Theta$, the transformed test-vectors can be generally written as

$$\underline{z}_u = (0, 0, \ldots, 0, z_{j_k}^{(u)}, \ldots, z_{j_{n-1}}^{(u)}). \tag{25}$$

With the first k symbols being zero, it allows the entry size of the module to be reduced as described in the following

C. Module Formulation and Minimisation

The MM interpolation will now be performed for each of the transformed test-vectors $\underline{z}_u$. With $\underline{z}_u$, polynomial $R(x)$ of (11) can be rewritten as

$$R(x) = \sum_{j=0}^{n-1} z_j^{(u)} L_j(x). \tag{26}$$

With $z_j^{(u)} = 0, \forall\ j \in \Theta$, it can be realised that

$$V(x) = \prod_{j \in \Theta} (x - \alpha_j) \tag{27}$$

is the gcd for both $G(x)$ of (10) and $R(x)$ of (26). Therefore, given a particular test-vector $\underline{z}_u$, we define

$$\widetilde{G}(x) = \frac{G(x)}{V(x)} = \prod_{j \in \overline{\Theta}} (x - \alpha_j), \tag{28}$$

$$\widetilde{R}(x) = \frac{R(x)}{V(x)} = \sum_{j \in \overline{\Theta}} \frac{z_j^{(u)}}{\varpi_j} \prod_{i \in \overline{\Theta}, i \neq j} (x - \alpha_i), \tag{29}$$

where $\varpi_j = \prod_{i=0, i \neq j}^{n-1} (\alpha_j - \alpha_i)$.

To continue, the following lemma is needed.

Lemma 2: Given a test-vector in the form of (25) and interpolation multiplicity of $m, V(x)^m | P_t(x, yV(x))$ holds.

Proof: With the generators defined by (12) and (13), when $0 \leq t \leq m, P_t(x, yV(x))$ can be expanded as

$$G^{m-t}(-R)^t + \binom{t}{1} G^{m-t}(-R)^{t-1} V y + \cdots + G^{m-t}(Vy)^t.$$

When $m < t \leq l, P_t(x, yV(x))$ can be elaborated as

$$(-R)^m (Vy)^{t-m} + \binom{m}{1} (-R)^{m-1} (Vy)^{t-m+1} + \cdots + (Vy)^t.$$

By acknowledging $V(x)|G(x)$ and $V(x)|R(x)$, it is straightforward to conclude that $V(x)^m | P_t(x, yV(x))$ for all t. ■

Armed with the above lemma, the following mapping can be performed on module generators $P_t(x, y)$

$$V(x)^{-m} P_t(x, yV(x)) \mapsto \widetilde{P}_t(x, y). \tag{30}$$

It results in the module generators becoming

$$\widetilde{P}_t(x, y) = \widetilde{G}(x)^{m-t} (y - \widetilde{R}(x))^t,\ \text{if } 0 \leq t \leq m, \tag{31}$$

$$\widetilde{P}_t(x,y) = (yV(x))^{t-m}(y-\widetilde{R}(x))^m, \text{ if } m < t \le l. \quad (32)$$

Since $\widetilde{P}_t(x,y) = \sum_{\tau \le t} \widetilde{P}_t^{(\tau)}(x)y^\tau$ and $\widetilde{P}_t^{(\tau)}(x) \in \mathbb{F}_q[x]$, we can generate an isomorphism of $\mathcal{M}_{m,l}$ and denote it as $\varphi(\mathcal{M}_{m,l})$. In $\varphi(\mathcal{M}_{m,l})$, its entry of row-$t$ column-τ is $\widetilde{P}_t^{(\tau)}(x)$. Note that $\deg \widetilde{P}_t^{(\tau)}(x) < \deg P_t^{(\tau)}(x)$, and it will lead to a simpler following row reduction process.

After the re-encoding transform, polynomials of $\mathbb{F}_q[x,y]$ are ordered under the $(1,-1)$-weighted degree. However, performing $\mathcal{A}_{m,l} = \varphi(\mathcal{M}_{m,l}) \cdot \mathcal{D}_{-1,l}$ will cause some of the entries leaving $\mathbb{F}_q[x]$. Instead, the following mapping will be performed before deploying the MS algorithm

$$\mathcal{A}_{m,l} = \varphi(\mathcal{M}_{m,l}) \cdot \ddot{\mathcal{D}}_{1,l}. \quad (33)$$

For a row $\underline{\xi}_t$ in $\mathcal{A}_{m,l}$, we have $\deg \underline{\xi}_t = \deg_{1,-1} \widetilde{P}_t(x,y) + l$. Afterwards, the MS algorithm can be applied on $\mathcal{A}_{m,l}$, delivering $\mathcal{A}'_{m,l}$ in the weak Popov form. By performing

$$\varphi(\mathcal{M}'_{m,l}) = \mathcal{A}'_{m,l} \cdot \ddot{\mathcal{D}}_{-1,l}, \quad (34)$$

polynomial $\widetilde{Q}$ can be taken from the row of $\varphi(\mathcal{M}'_{m,l})$ which corresponds to the minimal row of $\mathcal{A}'_{m,l}$. Finally, the minimal interpolated polynomial Q can be restored by

$$Q(x,y) = \sum_{\tau=0}^{l} \widetilde{Q}^{(\tau)}(x)V(x)^m \Big(\frac{y}{V(x)}\Big)^\tau. \quad (35)$$

It interpolates points $(\alpha_j, z_j^{(u)})$ for all j with a multiplicity of m. Factorisation [16] can be performed to retrieve the y-roots of Q, yielding $f'(x)$ in the form of (1), the intended message polynomial $\hat{f}(x)$ can now be estimated by

$$\hat{f}(x) = f'(x) + H(x). \quad (36)$$

Summarising the above description, the ACD-MM algorithm is stated as follows.

Algorithm 2 The ACD-MM Algorithm

Input: Π, η, m;
Output: $\hat{f}(x)$;
1: Determine metrics γ_j as in (21) and define Θ and $\overline{\Theta}$;
2: Formulate 2^η test-vectors $\underline{r}_u$ as in (22);
3: Perform re-encoding transform for all $\underline{r}_u$ as in (24);
4: For each transformed test-vector $\underline{z}_u$ **do**
5: Formulate $\varphi(\mathcal{M}_{m,l})$ using the generators (31)(32);
6: Perform Algorithm 1, yielding $\varphi(\mathcal{M}'_{m,l})$;
7: Restore the interpolated polynomial Q as in (35);
8: Factorise Q and estimate $\hat{f}$ as in (36);
9: End for

IV. Performance and Complexity Analysis

This section presents the decoding and complexity performance of the ACD-MM algorithm. The complexity is measured as the average number of arithmetic finite field operations in decoding a codeword frame. Using BPSK modulation, frame error rate (FER) performance over the additive white Gaussian noise (AWGN) channel is shown.

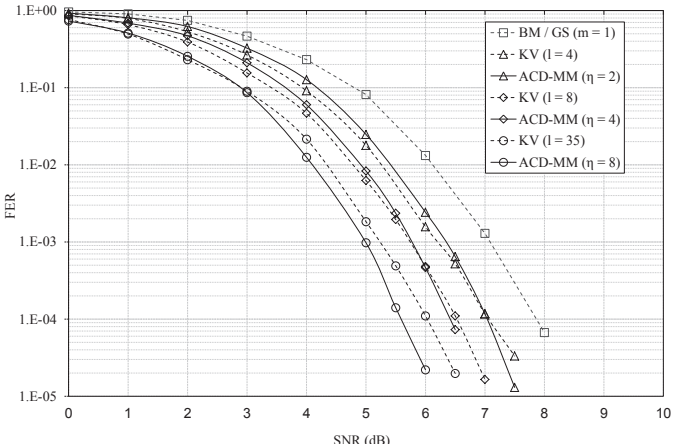

Fig. 1. Performance of the (15, 7) RS code over the AWGN channel.

Fig. 1 compares the ACD-MM algorithm with the BM, GS and KV algorithms for the (15, 7) RS code. The ACD-MM algorithm performs each Chase decoding trial with $m = 1$. In both the GS and KV algorithms, Koetter's interpolation is used. In particular, we compare the ACD-MM and KV algorithms under the condition that both of the soft decodings incur a similar interpolation cost (the total number of interpolation constraints) since it is a good indicator for the complexity of Koetter's interpolation. For the KV algorithm, it is shown in Definition 3 of [3]. For the ACD-MM algorithm with parameters η and m, the interpolation cost will be

$$2^\eta \cdot n \cdot \binom{m+1}{2}.$$

With $m = 1$ and $\eta = 2, 4$ and 8, the interpolation cost for ACD-MM decoding of the (15, 7) RS code will be 60, 240 and 3840, respectively. KV decoding with $l = 4, 8$ and 35 would incur a similar interpolation cost as the above ones. Fig. 1 shows that with a similar interpolation cost, the ACD-MM decoding can outperform the KV decoding, especially with a large η value. However, the ACD-MM algorithm is much simpler. This is evidenced by Table I that compares the complexity of several RS decoding algorithms.

With the same η, the ACD-MM algorithm performs the same as the LCC algorithm [7] since they adopt the same test-vector formulation. Table I shows that the LCC algorithm is less complex. However, the LCC algorithm is restricted with an interpolation multiplicity of one, limiting the error-correction capability of each Chase decoding trial. This will not be the case for the ACD-MM algorithm. Moreover, the LCC algorithm performs its Koetter's interpolation in a *binary tree growing* manner which benefits a lower complexity. But on the other hand, it prevents the Chase decoding trials to be performed in parallel. In contrast, the ACD-MM algorithm can perform its Chase decoding trials in parallel, benefiting a lower decoding latency.

Fig. 2 shows the ACD-MM decoding performance over the AWGN channel with different interpolation multiplicities. For the (15, 7) RS code, with $m = 1$, each Chase decoding

TABLE I
DECODING COMPLEXITY FOR THE (15, 7) RS CODE

BM	GS $(m = 1)$	KV $(l = 8)$	LCC $(\eta = 4)$	ACD-MM $(m = 1, \eta = 4)$	ACD-MM $(m = 4, \eta = 4)$
2.54×10^3	5.03×10^3	2.23×10^6	1.25×10^4	2.74×10^4	5.22×10^6

TABLE II
COMPLEXITY IMPACT OF THE RE-ENCODING TRANSFORM ON ACD-MM DECODING OF THE (15, 7) RS CODE

Parameters	$m = 1, \eta = 2$	$m = 1, \eta = 4$	$m = 1, \eta = 8$	$m = 4, \eta = 2$	$m = 4, \eta = 4$	$m = 4, \eta = 8$
w/o re-enc.	8.24×10^3	3.40×10^4	5.71×10^5	1.56×10^6	7.79×10^6	1.68×10^8
w re-enc.	8.74×10^3	2.74×10^4	4.32×10^5	1.05×10^6	5.22×10^6	1.26×10^8

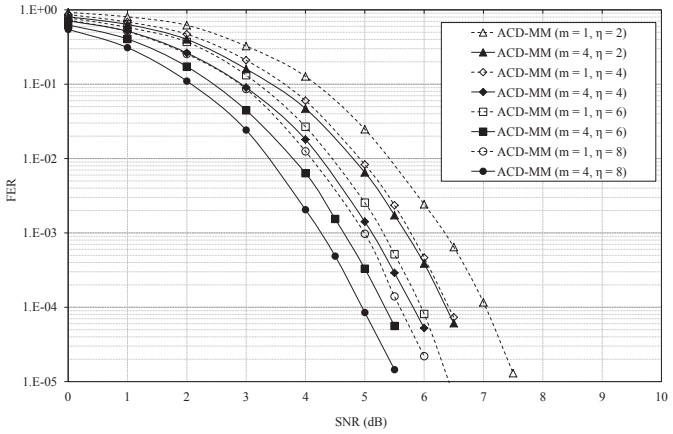

Fig. 2. Performance of ACD-MM decoding of the (15, 7) RS code with $m = 1$, 4 and $\eta = 2, 4, 6, 8$.

trial can correct four symbol errors. When $m = 4$, its error-correction capability is enhanced to five. With the same η value, this rewards 0.5 - 0.6 dB performance gain at the FER of 10^{-4}. Of course, such a performance gain is exchanged with a more complex decoding as indicated by Table I.

Finally, Table II shows the complexity impact brought by the re-encoding transform. Without re-encoding, the ACD-MM algorithm will perform GS list decoding for each formulated interpolation test-vector as described in Section II.B. It shows more complexity can be reduced with a larger interpolation multiplicity. E.g., when $\eta = 4$, performing re-encoding with $m = 1$ can reduce the complexity by a factor of $1/5$. With $m = 4$, the complexity reduction factor is $1/3$. However, notice that when $m = 1$ and $\eta = 2$, performing the re-encoding transform incurs a slight complexity increase. This is because the complexity reduction for MM interpolation has been offset by the extra computation brought by the transform itself. Finally, it should be pointed out that without the re-encoding as the KV algorithm, the ACD-MM $(m = 1, \eta = 4)$ still remains much simpler than the KV $(l = 8)$.

V. CONCLUSIONS

This paper has proposed the ACD-MM algorithm that achieves high decoding performance with low complexity and latency for RS codes. Facilitated by the re-encoding transform, the MM technique is utilised to solve the interpolation problem. Our simulation results show that it can outperform a number of RS decoding algorithms with a low computation.

ACKNOWLEDGEMENT

This work is sponsored by the National Natural Science Foundation of China (NSFC) with project ID 61372079 and the National Basic Research Program of China (973 Program) with project ID 2012CB316100.

REFERENCES

[1] J. Massey, "Shift register synthesis and BCH decoding," *IEEE Trans. Inform. Theory*, vol. 15, no. 1, pp. 122-127, Jan. 1991.
[2] V. Guruswami and M. Sudan, "Improved decoding of Reed-Solomon and algebraic-geometric codes," *IEEE Trans. Inform. Theory*, vol. 45, no. 6, pp. 1757-1767, Sept. 1999.
[3] R. Koetter and A. Vardy, "Algebraic soft-decision decoding of Reed-Solomon codes," *IEEE Trans. Inform. Theory*, vol. 49, no. 11, pp. 2809-2825, Nov. 2003.
[4] R. Koetter, "On algebraic decoding of algebraic-geometric and cyclic codes," Ph.D dissertation, Univ. Linköping, Linköping, Sweden, 1996.
[5] R. Koetter, J. Ma, and A. Vardy, "The re-encoding transformation in algebriac list-decoding of Reed-Solomon codes," *IEEE Trans. Inform. Theory*, vol. 57, no. 2, pp. 633-647, Feb. 2011.
[6] Y. Wu, "New list decoding algorithm for Reed-Solomon and BCH codes," *IEEE Trans. Inform. Theory*, vol. 54, no. 8, pp. 3611-3630, Aug. 2008.
[7] J. Bellorado and A. Kavčić, "Low-complexity soft-decoding algorithms for Reed-Solomon codes—Part I: An algebraic soft-in hard-out Chase decoder," *IEEE Trans. Inform. Theory*, vol. 56, no. 3, pp. 68-79, Mar. 2010.
[8] Y. Cassuto, J. Bruck, and R. McEliece, "On average complexity of Reed-Solomon list decoders," *IEEE Trans. Inform. Theory*, vol. 59, no. 4, pp. 2336-2351, Apr. 2013.
[9] L. Chen, S. Tang, and X. Ma, "Progressive algebraic soft-decision decoding of Reed-Solomon codes," *IEEE Trans. Commun.*, vol. 61, no. 2, pp. 433-442, Feb. 2013.
[10] K. Lee and M. O'Sullivan, "List decoding of Reed-Solomon codes from a Gröbner basis perspective," *J. Symb. Comput.*, vol. 43, no. 9, pp. 645-658, Sept. 2008.
[11] T. Mulders and A. Storjohann, "On lattice reduction for polynomial matrices," *J. Symb. Comput.*, vol. 35, no. 4, pp. 377-401, Apr. 2003.
[12] 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, Jul. 2005.
[13] P. Beelen and K. Brander, "Key equations for list decoding of Reed-Solomon codes and how to solve them," *J. Symb. Comput.*, vol. 45, no. 7, pp. 773-786, Jul. 2010.
[14] J. Nielsen and A. Zeh, "Multi-trial Guruswami-Sudan decoding for generalised Reed-Solomon codes," *Des. Codes Cryptogr.*, vol. 73, no. 2, pp. 507-527, Nov. 2014.
[15] M. Mohamed and M. Bossert, "A Chase-like decoding algorithm for Reed-Solomon codes based on the extended Euclidean algorithm," *Proc. 10th Int. ITG Conf. on Syst., Commun. and Coding (SCC)*, Hamburg, Germany, Feb. 2015.
[16] R. Roth and G. Ruckenstein, "Efficient decoding of Reed-Solomon codes beyond half the minimum distance," *IEEE Trans. Inform. Theory*, vol. 46, no. 1, pp. 246-256, Jan. 2000.