Interpolation Based Progressive Algebraic Chase Decoding of Reed-Solomon Codes

Jiancheng Zhao †, Li Chen †, Xiao Ma †, Martin Johnston ‡
† School of Information Science and Technology, Sun Yat-sen University, Guangzhou, China
‡ School of Electrical and Electronic Engineering, Newcastle University, Newcastle-upon-Tyne, UK
Email: zhaojch2@mail2.sysu.edu.cn, chenli55@mail.sysu.edu.cn, maxiao@mail.sysu.edu.cn, martin.johnston@ncl.ac.uk

***Abstract*—This paper proposes an interpolation based progressive algebraic Chase decoding (PACD) algorithm for Reed-Solomon (RS) codes. Based on the received information, 2^η $(\eta > 0)$ interpolation test-vectors are constructed. They are ordered using a reliability function, assessing their potential of yielding the intended message. The decoding is performed progressively granting priority to decode the test-vectors that are more likely to yield the intended message, and it will be terminated once the intended message is found. In the proposal, the decoding of a later test-vector utilizes the interpolation information that is generated during the decoding of the earlier ones. It results in the binary tree that represents the evolution of the interpolated polynomial sets growing in a *depth-first-search* manner. The PACD algorithm has the advantage of adapting its decoding computation to the channel condition, leveraging the average decoding complexity. This channel dependent feature will be validated by our simulation results which show that the PACD algorithm is less complex than various interpolation based algebraic decoding algorithms. We will also demonstrate that it can achieve a high RS decoding performance.**

***Index Terms*—Algebraic Chase decoding, complexity reduction, progressive decoding, Reed-Solomon codes.**

I. Introduction

Reed-Solomon (RS) codes are widely used in storage and communication systems. The conventional decoding algorithms [1] produce a unique decoded message, with error-correction capability limited by the half Hamming distance bound. The Guruswami-Sudan (GS) [2] algebraic list decoding algorithm was proposed to correct errors beyond the half distance bound by performing a curve-fitting process that contains two major steps, the interpolation and the factorization. The Koetter-Vardy (KV) soft-decision list decoding algorithm [3] was later proposed by introducing a reliability transform front-end. It significantly outperforms the GS and the unique decoding algorithms with a polynomial-time complexity.

However, the complexity of the algebraic list decoding algorithms remains much higher than that of the unique decoding algorithms. This is mainly caused by the interpolation that is an iterative polynomial construction process. To reduce the interpolation complexity, the GS decoding has been formulated as a rational curve-fitting problem utilizing the Berlekamp-Massey (BM) decoding outcomes by [4], resulting in a reduced interpolation multiplicity. The re-encoding approach [5] reduces the complexity by transforming the interpolation points into having zero y-coordinates. Consequently, a large part of iterative polynomial construction can be replaced by the polynomial initialization. The low-complexity Chase (LCC) decoding [6] reduces the complexity by exploiting the similarity among several interpolation test-vectors. It is shown to outperform the KV decoding with less computational cost. By arranging the test-vectors in a way such that the adjacent test-vectors have only one different point, the backward-forward (BF) interpolation can be applied, resulting in a hardware friendly BF-LCC decoding [7]. Moreover, a tree-based Chase decoding approach that better defines the test-vector set was recently reported in [8]. In order to adapt the algebraic decoding computation to the quality of the received information, the progressive algebraic soft decoding [9] was recently proposed. By progressively enlarging the designed factorization output list size (OLS), it deploys the least algebraic decoding effort in finding the intended message. In a good channel condition, most of the decoding events can deliver the intended message with a small OLS value. As a result, the average complexity of multiple decoding events can be reduced. Similarly, an interpolation algorithm that adjusts its computation to the number of instantaneous errors was proposed in [10].

In order to reduce the average decoding complexity while maintain a high RS decoding performance, this paper proposes the progressive algebraic Chase decoding (PACD) algorithm which is empowered by both the progressive decoding and the LCC decoding. A reliability function is defined to assess each test-vector's potential of yielding the intended message. The test-vector with a higher potential is decoded prior to the less potential one, and the decoding will be terminated once the intended message is found. The proposed algorithm enables the message to be found at an earlier Chase decoding stage, saving efforts that would otherwise be deployed to decode all the test-vectors. Consequently, average complexity of the PACD algorithm can be reduced by improving the channel condition. This channel dependent feature will be analyzed and validated by our simulation results. We show that the PACD algorithm is less complex than most of the interpolation based algebraic decoding algorithms with maintaining a high RS decoding performance. Its potential applications can be found in RS coded data communication systems.

II. Preliminaries

Let $\mathbb{F}_q$ be the finite field of size q and it is denoted as $\mathbb{F}_q = \{0, 1, 2, \ldots, q-1\}$ for convenience. Let $\mathbb{F}_q[x, y]$ denote the bivariate polynomial ring defined over $\mathbb{F}_q$. We consider

an (n, k) RS code, where n $(n = q - 1)$ and k $(k < n)$ are the length and the dimension of the code, respectively. Given a message vector $\overline{m} = (m_0, m_1, \ldots, m_{k-1}) \in \mathbb{F}_q^k$, the corresponding message polynomial can be written as

$$m(x) = \sum_{u=0}^{k-1} m_u x^u. \tag{1}$$

The RS encoding can be realized by

$$\overline{c} = (c_0, c_1, \ldots, c_{n-1}) = (m(x_0), m(x_1), \ldots, m(x_{n-1})), \tag{2}$$

where $\{x_0, x_1, \ldots, x_{n-1}\} \subseteq \mathbb{F}_q \setminus \{0\}$.

The PACD algorithm organizes monomials $x^a y^b$ using the (1, -1)-lexicographic order (ord) [6]. Given a polynomial $Q = \sum Q_{ab} x^a y^b \in \mathbb{F}_q[x, y]$, if $x^{a'} y^{b'}$ is the leading monomial with coefficient $Q_{a'b'} \neq 0$, its leading order is $\mathrm{lod}(Q) = \mathrm{ord}(x^{a'} y^{b'})$. For two polynomials $(Q, Q') \in \mathbb{F}_q[x, y]$, it holds that $Q < Q'$, if $\mathrm{lod}(Q) < \mathrm{lod}(Q')$.

III. The PACD Algorithm

The PACD algorithm constructs a set of interpolation test-vectors which will be ordered according to their potential of yielding the intended message. For all the test-vectors, interpolation of the common elements is performed once and shared by them. Interpolation of the uncommon elements will then be performed progressively granting priority to decode the test-vectors that are more likely to yield the intended message.

A. Interpolation Test-Vectors and Their Ordering

Assuming an RS codeword $\overline{c}$ is transmitted over a discrete memoryless channel, and the received vector $\overline{\mathfrak{R}} = (\mathfrak{R}_0, \mathfrak{R}_1, \ldots, \mathfrak{R}_{n-1}) \in \mathbb{R}^n$ is observed. The reliability matrix $\mathbf{\Pi} \in \mathbb{R}^{q \times n}$ can be further obtained. By assuming the rows of matrix $\mathbf{\Pi}$ are indexed by the elements of $\mathbb{F}_q$, its entries $\pi_{i,j}$ are the *a posteriori* probabilities defined as

$$\pi_{i,j} = \Pr[c_j = i | \mathfrak{R}_j], \text{ for } i \in \mathbb{F}_q \text{ and } 0 \leq j \leq n-1. \tag{3}$$

Let $i_j^1 = \arg\max_{i \in \mathbb{F}_q} \{\pi_{i,j}\}$ and $i_j^2 = \arg\max_{i \in \mathbb{F}_q, i \neq i_j^1} \{\pi_{i,j}\}$ denote the row indices of the largest and the second largest entries of column j, respectively. The most and the second most likely hard-decision results of c_j are $y_j^1 = i_j^1$ and $y_j^2 = i_j^2$, respectively. The following metric is defined to assess the reliability of the received information for c_j

$$\gamma_j = \frac{\pi_{i_j^2, j}}{\pi_{i_j^1, j}} \tag{4}$$

and $\gamma_j \in [0, 1]$. With γ_j approaches 1, the received information is less reliable. While with γ_j approaches 0, the received information is more reliable. Sorting all the γ_j values in an ascending order yields a new symbol index sequence $j_0, j_1, \ldots, j_{k-1}, \ldots, j_{n-1}$, which indicates $\gamma_{j_0} < \gamma_{j_1} < \cdots < \gamma_{j_{k-1}} < \cdots < \gamma_{j_{n-1}}$. Let us define $\Theta = \{j_0, j_1, \ldots, j_{k-1}\}$ as the index set of the k most reliable symbols, and $\Theta^{\mathrm{c}} = \{j_k, j_{k+1}, \ldots, j_{n-1}\}$. Moreover, η least reliable symbols will be selected from Θ^{c} and they can be realized as either y_j^1 or y_j^2. Let $\Phi = \{j_{n-\eta}, j_{n-\eta+1}, \ldots, j_{n-1}\} \subseteq \Theta^{\mathrm{c}}$ denote the index set of the selected unreliable symbols, and $\Phi^{\mathrm{c}} = \{j_0, j_1, \ldots, j_{n-\eta-1}\}$. The interpolation test-vectors can be generally written as

$$\overline{\mathcal{Y}}_v = (\mathcal{Y}_{v,0}, \mathcal{Y}_{v,1}, \ldots, \mathcal{Y}_{v,n-1}), \tag{5}$$

where

$$\mathcal{Y}_{v,j} = \begin{cases} y_j^1, \text{ if } j \in \Phi^{\mathrm{c}}, \\ y_j^1 \text{ or } y_j^2, \text{ if } j \in \Phi. \end{cases} \tag{6}$$

Since there are two decisions for each of the η unreliable symbols, 2^η interpolation test-vectors will be constructed and $v = 1, 2, \ldots, 2^\eta$. By considering each test-vector consists of n received symbols, the following metric is defined to assess the reliability of the test-vectors

$$\Omega_v = \sum_{j=0}^{n-1} \log(\pi_{\mathcal{Y}_{v,j}, j}), \tag{7}$$

where the base of the logarithm is ten. A higher Ω_v value indicates test-vector $\overline{\mathcal{Y}}_v$ is more reliable and it has a higher potential of yielding the intended message. Therefore, all the 2^η test-vectors will be ordered according to their reliability function Ω_v, and the one that has a larger Ω_v value will be decoded earlier. Since Ω_v can also be written as

$$\Omega_v = \sum_{j \in \Phi^{\mathrm{c}}} \log(\pi_{\mathcal{Y}_{v,j}, j}) + \sum_{j \in \Phi} \log(\pi_{\mathcal{Y}_{v,j}, j}), \tag{8}$$

and all the reliability functions share a common part of $\sum_{j \in \Phi^{\mathrm{c}}} \log(\pi_{\mathcal{Y}_{v,j}, j})$, the ordering metric can be simplified into

$$\Omega_v' = \sum_{j \in \Phi} \log(\pi_{\mathcal{Y}_{v,j}, j}). \tag{9}$$

Sorting all the 2^η test-vectors according to their Ω_v' values yields a new test-vector index sequence $v_1, v_2, \ldots, v_{2^\eta}$, which indicates $\Omega_{v_1} > \Omega_{v_2} > \cdots > \Omega_{v_{2^\eta}}$. The PACD algorithm first decodes $\overline{\mathcal{Y}}_{v_1}$, then decodes $\overline{\mathcal{Y}}_{v_2}$, and etc. Note that $\overline{\mathcal{Y}}_{v_1}$ is the hard-decision received vector.

B. Common Element Interpolation

For a test-vector $\overline{\mathcal{Y}}_v$, interpolation is to construct a minimal polynomial $Q(x, y)$ that satisfies $Q(x_j, \mathcal{Y}_{v,j}) = 0$ for all j. Since all the interpolation test-vectors share the common symbols y_j^1 for $j \in \Phi^{\mathrm{c}}$, interpolation will be performed regarding the common elements first. Its outcome will be shared by all the test-vectors. In this paper, we will discuss the test-vectors $\overline{\mathcal{Y}}_v$ in the content of $\overline{\mathcal{Y}}_v = (\mathcal{Y}_{v,j_0}, \mathcal{Y}_{v,j_1}, \ldots, \mathcal{Y}_{v,j_{n-1}})$. The interpolation order for the n points is $(x_{j_0}, \mathcal{Y}_{v,j_0}) \rightarrow (x_{j_1}, \mathcal{Y}_{v,j_1}) \rightarrow \cdots \rightarrow (x_{j_{n-1}}, \mathcal{Y}_{v,j_{n-1}})$.

The common element interpolation is assisted by the re-encoding transform. The re-encoding polynomial is defined as

$$\Psi(x) = \sum_{j \in \Theta} y_j^1 \psi_j(x), \tag{10}$$

where $\psi_j(x) = \prod_{(j,\delta) \in \Theta, j \neq \delta} \frac{x - x_\delta}{x_j - x_\delta}$. The re-encoding polynomial implies $\Psi(x_j) = y_j^1$ for $j \in \Theta$. Therefore, given a

test-vector $\overline{\mathcal{Y}}_v$, by performing the re-encoding transform

$$\mathcal{Y}'_{v,j} = \mathcal{Y}_{v,j} - \Psi(x_j) \tag{11}$$

for all of its entries, it can be transformed into $\overline{\mathcal{Y}}'_v = (0, \ldots, 0, \mathcal{Y}'_{v,j_k}, \ldots, \mathcal{Y}'_{v,j_{n-1}})$. Interpolation for points $(x_j, 0)$ where $j \in \Theta$ can be done by

$$V(x) = \prod_{j \in \Theta} (x - x_j). \tag{12}$$

Interpolation can then start by initializing the following polynomial set $\mathcal{G}^* = \{g_1^*(x,y), g_2^*(x,y)\} = \{V(x), y\}$. Since $\mathcal{G}^* = \{V(x) \cdot 1, V(x) \cdot \frac{y}{V(x)}\}$, the initialized polynomial set can be further simplified into

$$\mathcal{G} = \{g_1(x,y), g_2(x,y)\} = \{1, y\}, \tag{13}$$

and the remaining interpolation points $(x_j, \mathcal{Y}'_{v,j})$ for $j \in \Theta^{\mathrm{c}}$ are transformed into

$$(x_j, \mathcal{Y}''_{v,j}) = \left(x_j, \frac{\mathcal{Y}'_{v,j}}{V(x_j)}\right). \tag{14}$$

After the re-encoding transform, polynomials of $\mathcal{G}$ will further interpolate points $(x_j, \mathcal{Y}''_{v,j})$ for $j \in \Theta^{\mathrm{c}}$.

Since $\eta \leqslant n - k$, the test-vectors can share more than k common symbols. Let us define $\mathrm{A}_c = \Theta^{\mathrm{c}} \cap \Phi^{\mathrm{c}} = \{j_k, j_{k+1}, \ldots, j_{n-\eta-1}\}$ and $\mathrm{A}_u = \Theta^{\mathrm{c}} \cap \Phi = \{j_{n-\eta}, j_{n-\eta+1}, \ldots, j_{n-1}\}$. In the common element interpolation, polynomials of $\mathcal{G}$ will interpolate points $(x_j, \mathcal{Y}''_{v,j})$ for $j \in \mathrm{A}_c$. For $(x_j, \mathcal{Y}''_{v,j})$, the polynomials' interpolation property can be judged by $g_1(x_j, \mathcal{Y}''_{v,j})$ and $g_2(x_j, \mathcal{Y}''_{v,j})$ [2]. Let

$$f(x,y) = \min\{g_i(x,y) \in \mathcal{G} | g_i(x_j, \mathcal{Y}''_{v,j}) \neq 0 \text{ and } i = 1,2\}, \tag{15}$$

and

$$g(x,y) = \mathcal{G} \setminus \{f(x,y)\}. \tag{16}$$

By performing the following bilinear modifications

$$g'(x,y) = g(x,y) - \frac{g(x_j, \mathcal{Y}''_{v,j})}{f(x_j, \mathcal{Y}''_{v,j})} \cdot f(x,y), \tag{17}$$

$$f'(x,y) = (x - x_j) \cdot f(x,y), \tag{18}$$

the updated polynomials would interpolate the point. Repeating the above process for all the points that are indexed by A_c, an updated polynomial set $\widetilde{\mathcal{G}}$ will be obtained and

$$\widetilde{\mathcal{G}} = \{g_i(x,y) | g_i(x_j, \mathcal{Y}''_{v,j}) = 0, \ \forall\ j \in \mathrm{A}_c \text{ and } i = 1,2\}. \tag{19}$$

It will be utilized by the following progressive uncommon element interpolation.

C. Progressive Uncommon Element Interpolation

The progressive uncommon element interpolation is first performed for test-vector $\overline{\mathcal{Y}}'_{v_1}$ by interpolating points $(x_j, \mathcal{Y}''_{v_1,j})$ for $j \in \mathrm{A}_u$. If the intended message can be found by factorizing the interpolation outcome, the decoding will be terminated. Otherwise, it will be further performed for test-vectors $\overline{\mathcal{Y}}'_{v_2}$, $\overline{\mathcal{Y}}'_{v_3}$, and etc. For a test-vector $\overline{\mathcal{Y}}'_v$, we define

$$\begin{aligned} \mathcal{G}_v^{(\tau)} \quad = \quad & \{g_i(x,y) | g_i(x_j, \mathcal{Y}''_{v,j}) = 0, \text{ for } j = j_k, \\ & j_{k+1}, \ldots, j_{n-\eta+\tau-1} \text{ and } i = 1,2\}. \end{aligned} \tag{20}$$

It is the polynomial set that has further interpolated τ points $(x_j, \mathcal{Y}''_{v,j})$ that are indexed by A_u and $0 \leqslant \tau \leqslant \eta$. When $\tau = 0$, $\mathcal{G}_v^{(0)} = \widetilde{\mathcal{G}}$ as all the test-vectors inherit the the common element interpolation result. With this notation, the PACD algorithm generates the polynomial sets in the following order: $\mathcal{G}_{v_1}^{(1)} \to \mathcal{G}_{v_1}^{(2)} \to \cdots \to \mathcal{G}_{v_1}^{(\eta)}$, $\mathcal{G}_{v_2}^{(1)} \to \mathcal{G}_{v_2}^{(2)} \to \cdots \to \mathcal{G}_{v_2}^{(\eta)}$, …, $\mathcal{G}_{v_{2^\eta}}^{(1)} \to \mathcal{G}_{v_{2^\eta}}^{(2)} \to \cdots \to \mathcal{G}_{v_{2^\eta}}^{(\eta)}$. In general, those polynomial sets can be denoted as $\mathcal{G}_{v_\varpi}^{(\tau)}$, where $\varpi = 1, 2, \ldots, 2^\eta$. If all the 2^η test-vectors are interpolated, $\eta \cdot 2^\eta$ polynomial sets will be generated. The following proposition shows that they are not unique and the PACD algorithm reduces the decoding computation utilizing this property.

Proposition 1: For any two test-vectors $\overline{\mathcal{Y}}'_{v_\varpi}$ and $\overline{\mathcal{Y}}'_{v_{\varpi'}}$, if $\mathcal{Y}''_{v_\varpi,j} = \mathcal{Y}''_{v_{\varpi'},j}$ for $j = j_{n-\eta}, j_{n-\eta+1}, \ldots, j_{n-\eta+\tau-1}$, then $\mathcal{G}_{v_\varpi}^{(1)} = \mathcal{G}_{v_{\varpi'}}^{(1)}$, $\mathcal{G}_{v_\varpi}^{(2)} = \mathcal{G}_{v_{\varpi'}}^{(2)}$, …, $\mathcal{G}_{v_\varpi}^{(\tau)} = \mathcal{G}_{v_{\varpi'}}^{(\tau)}$.

Without referring to the test-vector ordering, polynomial set $\mathcal{G}_v^{(\tau)}$ is obtained by interpolating polynomials of $\mathcal{G}_v^{(\tau-1)}$ for either of the following two points

$$P_\tau^1 = \left(x_{j_{n-\eta+\tau-1}}, \frac{y^1_{j_{n-\eta+\tau-1}} - \Psi(x_{j_{n-\eta+\tau-1}})}{V(x_{j_{n-\eta+\tau-1}})}\right),$$

$$P_\tau^2 = \left(x_{j_{n-\eta+\tau-1}}, \frac{y^2_{j_{n-\eta+\tau-1}} - \Psi(x_{j_{n-\eta+\tau-1}})}{V(x_{j_{n-\eta+\tau-1}})}\right).$$

If all the test-vectors are interpolated, the evolution of the polynomial sets $\mathcal{G}_v^{(\tau)}$ with τ growing from zero to η can be illustrated as a binary tree that is shown by Fig. 1. At layer τ, there are 2^τ distinct polynomial sets $\mathcal{G}_v^{(\tau)}$, each of which is shared by $2^{\eta-\tau}$ test-vectors. A complete path from $\mathcal{G}_v^{(0)}$ to $\mathcal{G}_v^{(\eta)}$ indicates the uncommon element interpolation for a particular test-vector $\overline{\mathcal{Y}}'_v$. The LCC algorithm [6] grows the binary tree in a *layer-by-layer* manner. Polynomial sets $\mathcal{G}_v^{(\tau)}$ of layer τ are fully determined by interpolating the polynomial sets $\mathcal{G}_v^{(\tau-1)}$ of layer $\tau - 1$. Finally, 2^η polynomial sets $\mathcal{G}_v^{(\eta)}$ will be generated. They correspond to the 2^η test-vectors.

In contrast, the PACD algorithm grows the binary tree in a *depth-first-search* manner. It first generates a completed path from $\mathcal{G}_{v_1}^{(0)}$ to $\mathcal{G}_{v_1}^{(\eta)}$. If the intended message can be found by factorizing the minimal polynomial of $\mathcal{G}_{v_1}^{(\eta)}$, the decoding will be terminated. Otherwise, it generates another path from $\mathcal{G}_{v_2}^{(0)}$ to $\mathcal{G}_{v_2}^{(\eta)}$, and etc. Note that the maximum likelihood (ML) criterion that is stated as *Lemma 1* of [11] is utilized to validate the factorization output. However, based on *Proposition 1*, it can be realized that the interpolation of a later decoded test-vector $\overline{\mathcal{Y}}'_{v_\varpi}$ with $\varpi > 1$ does not necessary need to start from $\mathcal{G}_{v_\varpi}^{(0)}$. It can utilize the intermediate interpolation information that is obtained and memorized during decoding the previous $\varpi - 1$ test-vectors. The intermediate interpolation information

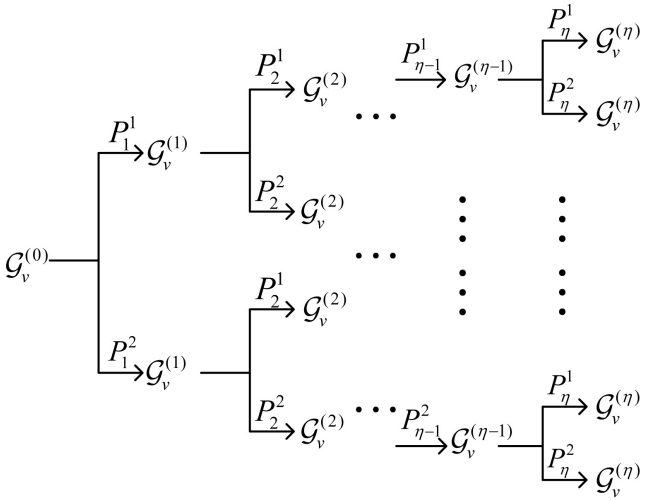

Fig. 1. Binary tree representation of the uncommon element interpolation.

is defined as

$$\mathcal{M}_{\varpi-1} = \{\mathcal{G}_{v_{\varpi'}}^{(\tau)}, \text{ for } 0 \leqslant \tau < \eta \text{ and } 1 \leqslant \varpi' \leqslant \varpi-1\}. \quad (21)$$

Note that $\mathcal{G}_{v_{\varpi'}}^{(\eta)}$ does not need to be memorized since it only corresponds to $\overline{\mathcal{Y}}'_{v_{\varpi'}}$. Hence, to decode a test-vector $\overline{\mathcal{Y}}'_{v_\varpi}$, we need to first assess its similarity with the decoded ones $\overline{\mathcal{Y}}'_{v_{\varpi'}}$ where $\varpi' < \varpi$. In the binary tree, let $\Xi_\varpi(\varpi')$ denote the number of layers shared by $\overline{\mathcal{Y}}'_{v_\varpi}$ and $\overline{\mathcal{Y}}'_{v_{\varpi'}}$ as

$$\begin{aligned}\Xi_\varpi(\varpi') \quad = \quad & \max\{\tau | \mathcal{Y}''_{v_\varpi, j_{n-\eta+\tau'-1}} = \mathcal{Y}''_{v_{\varpi'}, j_{n-\eta+\tau'-1}}, \\ & \text{for } 0 \leq \tau' \leq \tau\}. \end{aligned} \quad (22)$$

The decoded test-vectors $\overline{\mathcal{Y}}'_{v_{\varpi'}}$ that share the maximal number of layers with $\overline{\mathcal{Y}}'_{v_\varpi}$ should be identified and denoted as

$$\{\overline{\mathcal{Y}}'_{v_{\varpi^*}} : \ \varpi^* = \arg\max_{\varpi'<\varpi} \{\Xi_\varpi(\varpi')\}\}. \quad (23)$$

Note that there can be more than one decoded test-vector being identified. One of them will be selected and denoted as $\overline{\mathcal{Y}}'_{v_{\varpi*}}$. Recalling *Proposition 1*, we know that $\mathcal{G}_{v_\varpi}^{(\tau)} = \mathcal{G}_{v_{\varpi*}}^{(\tau)}$ for $\tau = 0, 1, \ldots, \Xi_\varpi(\varpi^*)$, while $\mathcal{G}_{v_{\varpi*}}^{(0)}$, $\mathcal{G}_{v_{\varpi*}}^{(1)}$, $\ldots$, and $\mathcal{G}_{v_{\varpi^*}}^{(\Xi_\varpi(\varpi^*))}$ are the memorized information of $\mathcal{M}_{\varpi-1}$. Therefore, to generate $\mathcal{G}_{v_\varpi}^{(\eta)}$, we will first initialize

$$\mathcal{G}_{v_\varpi}^{(\Xi_\varpi(\varpi^*))} = \mathcal{G}_{v_{\varpi^*}}^{(\Xi_\varpi(\varpi^*))}. \quad (24)$$

Polynomial set $\mathcal{G}_{v_\varpi}^{(\Xi_\varpi(\varpi^*))}$ will then interpolate points $(x_j, \mathcal{Y}''_{v_\varpi, j})$ for $j = j_{n-\eta+\Xi_\varpi(\varpi^*)}, \ldots, j_{n-1}$, yielding polynomial set $\mathcal{G}_{v_\varpi}^{(\eta)}$. The memorized information is updated into $\mathcal{M}_\varpi$. The minimal polynomial of $\mathcal{G}_{v_\varpi}^{(\eta)}$, i.e.,

$$\widetilde{Q}_{v_\varpi}(x, y) = \min\{g_i(x, y) \in \mathcal{G}_{v_\varpi}^{(\eta)}, \ i = 1, 2\}, \quad (25)$$

will be chosen. Since $\widetilde{Q}_{v_\varpi}(x, y) = \widetilde{q}_{v_\varpi,0}(x) + y \cdot \widetilde{q}_{v_\varpi,1}(x)$, and polynomial $V(x)$ has been extracted from set $\mathcal{G}^*$ at the beginning of the common element interpolation, the interpolated polynomial $Q_{v_\varpi}(x, y)$ should be reconstructed by

$$Q_{v_\varpi}(x, y) = \widetilde{q}_{v_\varpi,0}(x) \cdot V(x) + y \cdot \widetilde{q}_{v_\varpi,1}(x). \quad (26)$$

It satisfies $Q_{v_\varpi}(x_j, \mathcal{Y}'_{v_\varpi, j}) = 0$ for all j.

A message polynomial $m'(x)$ that is in the form of (1) can be obtained by factorizing $Q_{v_\varpi}(x, y)$ [12]. An estimation of the intended message polynomial is further generated by

$$\widehat{m}(x) = m'(x) + \Psi(x). \quad (27)$$

If the re-encoding of $\widehat{m}(x)$ yields an ML codeword, the decoding will be terminated. Otherwise, the next test-vector $\overline{\mathcal{Y}}_{v_{\varpi+1}}$ will be decoded based on the memorized information $\mathcal{M}_\varpi$. Summarizing the above descriptions, the PACD algorithm is presented as in Algorithm 1.

Algorithm 1 The PACD algorithm for RS code

Input: η;
Initialization: $\varpi = 1$.
1: Construct 2^η test-vectors $\overline{\mathcal{Y}}_v$;
2: Calculate Ω'_v as in (9) and order the 2^η test-vectors;
3: Perform the re-encoding transform as in (11);
4: Initialize the polynomial group $\mathcal{G}$ as in (13);
5: Interpolate points $(x_j, \mathcal{Y}''_{v,j})$ for $j \in \mathrm{A}_c$ as in (15)-(18), yielding $\widetilde{\mathcal{G}}$ of (19);
6: for test-vector $\overline{\mathcal{Y}}'_{v_\varpi}$ **do**
7: **if** $\varpi = 1$ **then**
8: Let $\mathcal{G}_{v_1}^{(0)} = \widetilde{\mathcal{G}}$;
9: Interpolate points $(x_j, \mathcal{Y}''_{v_1,j})$ for $j \in \mathrm{A}_u$ as in (15)-(18), yielding $\mathcal{G}_{v_1}^{(\eta)}$;
10: **else**
11: Determine $\Xi_\varpi(\varpi')$ as in (22);
12: Identify a decoded test-vector $\overline{\mathcal{Y}}'_{v_{\varpi^*}}$ as in (23);
13: Let $\mathcal{G}_{v_\varpi}^{(\Xi_\varpi(\varpi^*))} = \mathcal{G}_{v_{\varpi^*}}^{(\Xi_\varpi(\varpi^*))}$;
14: Interpolate points $(x_j, \mathcal{Y}''_{v_\varpi,j})$ for $j = j_{n-\eta+\Xi_\varpi(\varpi^*)}, \ldots, j_{n-1}$ as in (15)-(18), yielding $\mathcal{G}_{v_\varpi}^{(\eta)}$;
15: **end if**
16: Find the minimal polynomial $\widetilde{Q}_{v_\varpi}(x, y)$ of (25);
17: Restore polynomial $Q_{v_\varpi}(x, y)$ as in (26);
18: Factorize $Q_{v_\varpi}(x, y)$ to obtain $m'(x)$;
19: Estimate $\widehat{m}(x)$ as in (27);
20: Perform the re-encoding of $\widehat{m}(x)$;
21: **if** the codeword satisfies the ML criterion **then**
22: Output $\widehat{m}(x)$ and terminate the decoding;
23: **else**
24: Update $\varpi = \varpi + 1$ and go to step 6;
25: **end if**
26: end for

If none of the 2^η message candidates yields an ML codeword, the PACD algorithm results in a full growth of the binary tree. Among all the message candidates, the one that yields the most likely codeword will be selected. It should be pointed out that the proposed Chase decoding has a certain memory cost since its uncommon element interpolation requires the intermediate information to be memorized.

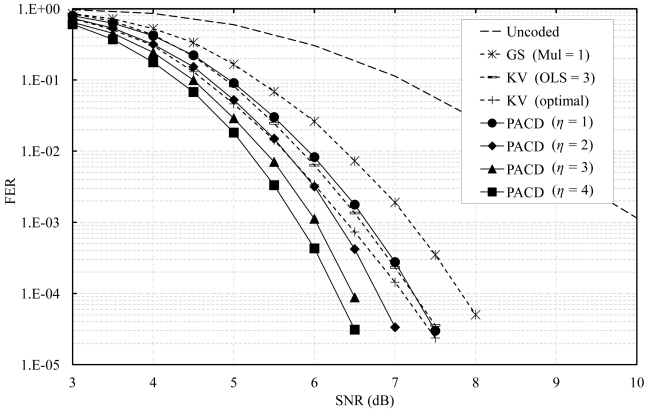

Fig. 2. Performance of the (31, 27) RS code over the AWGN channel.

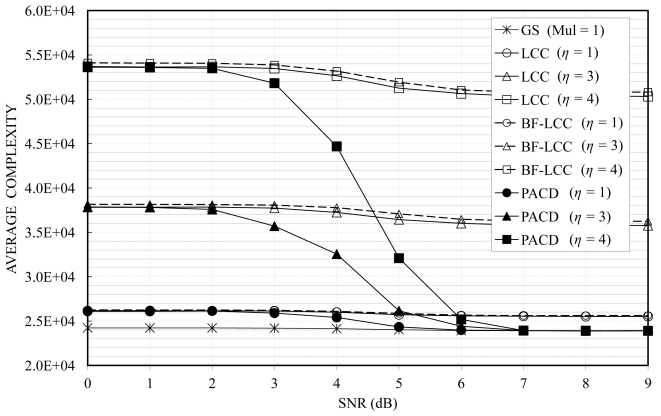

Fig. 3. Average complexity of the PACD algorithm in decoding the (31, 27) RS code.

IV. Performance and Complexity Analyses

This section presents the decoding performance and complexity analyses of the PACD algorithm. Decoding simulation is performed for the (31, 27) RS code over the additive white Gaussian noise (AWGN) channel using the BPSK modulation. The complexity is measured as the average number of finite field arithmetic operations in decoding an RS codeword, which is averaged over 10000 decoding events per signal-to-noise ratio (SNR). Performances of the interpolation based algebraic decoding algorithms, including the GS, KV, LCC and BF-LCC algorithms, are given as the comparison benchmarks. They are all facilitated by the re-encoding transform.

A. Performance Analysis

Fig. 2 shows the frame error rate (FER) performance of the (31, 27) RS code. By increasing η, more test-vectors are decoded and performance of the PACD algorithm can be improved. In particular, by increasing η from 1 to 4, 1 dB coding gains can be further achieved for the $(31, 27)$ RS code at the FER of 10^{-4}. With $\eta = 2$, the PACD algorithm starts to outperform both of the GS and the KV algorithms. The GS decoding is performed with an interpolation multiplicity of one and it reaches the hard-decision list decoding bound of $n - \lfloor\sqrt{n(k-1)}\rfloor - 1$. The KV decoding is performed with a designed factorization OLS of three. The theoretical optimal KV decoding performance is also presented. Note that with the same η, the PACD algorithm has the same decoding performance as the LCC and the BF-LCC algorithms.

B. Complexity Analysis

Since the PACD algorithm may terminate after decoding any one of the 2^η test-vectors, its average complexity is channel dependent. To further describe this channel dependent feature, we define the worst case decoding event as when all the 2^η test-vectors have been decoded, while the best case decoding event as when only one test-vector has been decoded. They correspond to a full growth of the binary tree and a growth of the root $\mathcal{G}_{v_1}^{(0)}$ to leaf $\mathcal{G}_{v_1}^{(\eta)}$ path, respectively. Let $\mathcal{C}_{\rm c}$ denote the complexity of the common element interpolation, and $\mathcal{C}_{\rm u}^{(1)}$ denote the average complexity[1] of determining a polynomial set $\mathcal{G}_v^{(\tau)}$ based on $\mathcal{G}_v^{(\tau-1)}$ during the uncommon element interpolation. Furthermore, let $\mathcal{C}_{\rm u}^{(2)}$ denote the complexity of factorization and the ML validation for each test-vector. In the worst case decoding event, there are $2(2^\eta - 1)$ polynomial sets to be determined and 2^η polynomials to be factorized. Hence, the PACD algorithm's complexity is

$$\mathcal{C}_{\rm worst} = \mathcal{C}_{\rm c} + 2(2^\eta - 1)\cdot\mathcal{C}_{\rm u}^{(1)} + 2^\eta\cdot\mathcal{C}_{\rm u}^{(2)}. \tag{28}$$

In the best case decoding event, there are only η polynomial sets to be determined and only one polynomial to be factorized. The PACD algorithm's complexity becomes

$$\mathcal{C}_{\rm best} = \mathcal{C}_{\rm c} + \eta\cdot\mathcal{C}_{\rm u}^{(1)} + \mathcal{C}_{\rm u}^{(2)}. \tag{29}$$

The above analysis shows that the worst case complexity grows exponentially with η and it is also the computational cost of the LCC algorithm. While in the best case, there is only one test-vector having been decoded. Its complexity will be the same as that of the GS decoding with a multiplicity of one. In order to validate this channel dependent feature, Fig.3 compares the complexity of the PACD algorithm and its prototypes, the LCC and the BF-LCC algorithms, as well as the GS algorithm. It can be seen that by increasing the SNR, the average complexity of the PACD algorithm can be reduced significantly. In contrast, the average complexity of the LCC and the BF-LCC algorithms is less sensitive to the channel condition, since they always terminate after decoding all the test-vectors. For the PACD algorithm, in the low SNR region ($\leq$ 2dB), the worst case decoding events dominate. Hence, its average complexity is similar to that of the LCC and the BF-LCC algorithms. While in the high SNR region ($\geq$ 7dB), the best case decoding events dominate. As a result,

[1]As τ grows, it is more complex to determine $\mathcal{G}_v^{(\tau)}$, since the size of the polynomials increases as the interpolation goes on.

TABLE I
COMPLEXITY COMPARISON IN DECODING THE (31, 27) RS CODE USING THE KV, PBF-LCC AND PACD ALGORITHMS

SNR / Algorithms	0dB	1dB	2dB	3dB	4dB	5dB	6dB	7dB	8dB
KV (OLS $= 3$)	275 461	254 616	220 558	185 225	153 855	135 423	119 186	106 254	98 031
PBF-LCC ($\eta = 4$)	54 107	54 083	53 933	52 398	45 324	32 356	25 301	24 028	23 924
PACD ($\eta = 4$)	53 621	53 593	53 491	51 817	44 691	32 086	25 180	23 925	23 912

its average complexity converges to that of the GS algorithm. Note that the BF-LCC algorithm is slightly more complex than the LCC algorithm. This extra difference is incurred its backward interpolation.

Table I further compares the average complexity of the PACD algorithm with the KV algorithm. It vindicates the PACD algorithm's simplicity. With $\eta = 4$, Fig. 2 shows that the PACD algorithm has 1dB coding gain over the KV algorithm at the FER of 10^{-4}. Hence, for the (31, 27) RS code, the PACD algorithm outperforms the KV algorithm with less computational cost. But it should be acknowledged that the worst case complexity of the PACD algorithm grows exponentially with η, while complexity of the KV algorithm is polynomial-time. By increasing η, one could expect the worst case complexity of the PACD algorithm will outweight the KV algorithm. However, it is confined that $\eta \leq n-k$. For the above mentioned RS code, $\eta \leq 4$ and the worst case complexity of the PACD algorithm remains favorable.

Since the BF-LCC algorithm also delivers the polynomial sets $\mathcal{G}_v^{(\eta)}$ one by one, the authors have further integrated it with the progressive decoding, resulting in the progressive BF-LCC (PBF-LCC) algorithm. Similarly, the PBF-LCC algorithm terminates once an ML codeword is found. It is ensured that the first test-vector to be decoded is the hard-decision received vector. Table I shows that its average complexity can also be reduced with an increased SNR. The PACD algorithm is less complex than the PBF-LCC algorithm. The reason has two folds. First, the PACD algorithm does not require the backward interpolation. Second, with a decoding order that is sorted by the reliability function Ω'_v, the PACD algorithm is able to find an ML codeword at an earlier Chase decoding stage. Our simulation statistics indicates that the earlier decoded test-vector is more likely to yield an ML codeword. This may not be the case for the PBF-LCC algorithm. However, the authors should acknowledge that the proposal offers an effective but yet to be an optimal Chase decoding order. This reminds as an open issue for future work.

Finally, as mentioned earlier, the PACD algorithm does have a greater memory consumption. By assuming each polynomial coefficient consumes one memory unit, memorizing all the intermediate nodes of the binary tree requires at most $4(2^{\eta}(n-k-1)+(\eta+2))$ memory units.

V. CONCLUSION

This paper has proposed an interpolation based PACD algorithm for RS codes. In the proposal, each test-vector will be sequentially decoded according to its potential of yielding the intended message. It will be terminated once the intended message is found. It can adapt the decoding computation to the channel condition, leveraging the average decoding complexity. Our simulation results have shown that the average complexity of the PACD algorithm is channel dependent and it is simpler than various interpolation based algebraic decoding algorithms. Its error-correction performance has demonstrated its capability of outperforming the GS and the KV algorithms. Therefore, the proposal is an effective solution for achieving high RS decoding performances.

ACKNOWLEDGMENT

This work is sponsored by the National Basic Research Program of China (973 program) with project ID 2012CB316100, the National Natural Science Foundation of China (NSFC) with project ID 61372079 and the Fundamental Research Funds for the Central Universities in China.

REFERENCES

[1] L. Welch and E. R. Berlekamp, "Error correction for algebraic block codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Ste. Jovite, Canada, Sept. 1983.

[2] V. Guruswami and M. Sudan, "Improved decoding of Reed-Solomon and algebraic-geometric codes," *IEEE Trans. Inf. 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. Inf. Theory*, vol. 49, no. 11, pp. 2809-2825, Nov. 2003.

[4] Y. Wu, "New list decoding algorithms for Reed-Solomon and BCH codes," *IEEE Trans. Inf. Theory*, vol. 54, no. 8, pp. 3611-3630, Aug. 2008.

[5] R. Koetter, J. Ma, and A. Vardy, "The re-encoding transformation in algebraic list-decoding of Reed-Solomon codes," *IEEE Trans. Inf. Theory*, vol. 57, no. 2, pp. 633-647, Feb. 2011.

[6] J. Bellorado and A. Kavcic, "Low-complexity soft-decoding algorithms for Reed-Solomon codes - Part I: an algebraic soft-in hard-out Chase decoder," *IEEE Trans. Inf. Theory*, vol. 56, no. 3, pp. 945-959, Mar. 2010.

[7] J. Zhu, X. Zhang, Z. Wang, "Backward interpolation architecture for algebraic soft-decision Reed-Solomon decoding," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 17, no. 11, pp. 1602-1615, 2009.

[8] S. Tang and X. Ma, "A new Chase-type soft-decision decoding algorithm for Reed-Solomon codes," available at http://arxiv.org/abs/1309.1555.

[9] L. Chen, S. Tang and X. Ma, "Progressive algebraic soft-decision decoding of Reed-Solomon codes," *IEEE Trans. Commum.*, vol. 61, no. 2, pp. 433-442, Feb. 2013.

[10] Y. Cassuto, J. Bruck and R. J. McEliece, "On the average complexity of Reed-Solomon list decoders," *IEEE Trans. Inf. Theory*, vol. 59, no. 4, pp. 2336-2351, Feb. 2013.

[11] T. Kaneko, T. Nishijima, H. Inazumi, and S. Hirasawa, "An efficient maximum-likelihood-decoding algorithm for linear block codes with algebraic decoder," *IEEE Trans. Inf. Theory*, vol. 40, no. 2, pp. 320-327, Mar. 1994.

[12] R. Roth and G. Ruckenstein, "Efficient decoding of Reed-Solomon codes beyond half the minimum distance," *IEEE Trans. Inf. Theory*, vol. 46, no. 1, pp. 246-257, Jan. 2000.