A Progressive Interpolation Approach for Guruswami-Sudan Algorithm

Jingwei Zhang
School of Mathematical Science,
Yangzhou University,
Yangzhou, China.
Email:zhjingw@gmail.com

Chang-An Zhao
School of Educational Software
and Computer Science,
Guangzhou University,
Guangzhou, China

Li Chen and Xiao Ma
School of Information Science
and Technology, Sun Yat-sen University,
Guangzhou, China

***Abstract*—In this paper, we present a progressive interpolation approach in Guruswami-Sudan (GS) decoding of Reed-Solomon (RS) codes. The objective of the interpolation is to construct the minimal polynomial $Q(x, y)$ by the progressive approach with increasing multiplicities until the roots of $Q(x, y)$ give the correct message. Then the error-correction capability can be adaptively obtained by assigning a suitable multiplicity according to the number of errors occurred in the channel. We present an efficient way to update the polynomial set utilizing the previous computational results in the interpolation step. It enables the decoder to adjust its decoding complexity to the needed level. Simulation results suggest that the average decoding complexity of GS algorithm can be significantly reduced by the progressive approach for RS codes.**

***Keywords*—Guruswami-Sudan algorithm, progressive interpolation, Reed-Solomon codes.**

I. Introduction

Reed-Solomon (RS) codes are one of the most important error-correcting codes that are widely applied in satellite communications, magnetic and optical data storage. An (n, k) RS code, defined over a finite field $\mathbb{F}_q$, achieves the maximum separable distance of $d = n - k + 1$, where n and k are the length and the dimension, respectively. The classical decoding algorithms, such as Berlekamp-Massey (BM) algorithm [1] [2], Euclidean algorithm [3] and Welch-Berlekamp (WB) algorithm [4], guarantee a unique codeword in the output provided that the number of errors is upper bounded by $\lfloor \frac{d-1}{2} \rfloor$. List decoding was introduced independently by Elias [5] and Wozenrcraft [6], which provided an idea for decoding beyond the half distance error-correction bounds. In [7], Sudan discovered a polynomial-time list decoding algorithm correcting at least $n - \sqrt{2nk}$ errors for the low-rate RS codes. Later, Guruswami and Sudan extended the list decoding algorithm by interpolating each point at least m times such that the error-correction capability can be improved up to $n - \sqrt{n(k-1)}$ for all rate RS codes [8].

The algebraic list decoding algorithm depends on the interpolation and factorization of polynomials. There has been a lot of work on complexity reduction of the algebraic list decoding algorithm [9]–[20]. Koetter *et.al* [9] [10] and Roth *et.al* [11] presented the efficient techniques to reduce the complexity of the interpolation and factorization, respectively. The progressive algebraic soft-decision decoding algorithm is devised for the RS codes by enlarging the list-size in [20]. From the hard-decision decoding point of view, the progressive list decoding algorithm is presented for adaptively obtaining the error-correction capability with increasing multiplicities. It is known that the interpolation process aims to construct the minimal polynomial $Q(x, y)$ by solving a system of homogeneous linear equations. This is the most computationally expensive step of GS algorithm. As the channel condition improves, for example, the Signal-to-Noise Ratio (SNR) increased, it is likely that less symbol errors will be introduced in most of the received word. Hence, it is not necessary to use a large multiplicity value for the interpolation since most of error patterns can be corrected by GS algorithm with a small multiplicity value. This inspires further research to adaptively assign a suitable multiplicity for the interpolation step according to the number of errors occurred in the channel.

Motivated by this observation, we propose a progressive interpolation approach of GS algorithm. The objective of the interpolation is to construct the minimal polynomial $Q(x, y)$ by the progressive approach with increasing multiplicities until the roots of $Q(x, y)$ give the correct message. We first set a small multiplicity m_1 for the interpolation such that the error-correction capability is equal to t_1. If the decoding output list is not empty, the decoder declares "*decoding success*". Otherwise, a suitable multiplicity $m_2 > m_1$ is assigned such that the error-correction capability improves to $t_2 \geq t_1+1$. The procedure can be repeated until the output list is not empty or the multiplicity is greater than a prescribed threshold value. An important advantage of our approach is that the interpolation can utilize the previous computational results. Due to the progressive approach, the average decoding complexity of the interpolation process can be reduced significantly for decoding the RS codes.

The rest of this paper is organized as follows. Section II gives a brief review of RS codes and GS algorithm. In Section III, the progressive interpolation approach is devised for RS codes. A complexity analysis and simulation results are presented in Section IV. Finally, Section V concludes this paper.

II. Preliminaries

Let $\mathbb{F}_q$ be a finite field of size q. An (n,k) RS code can be generated by evaluating the message polynomial $f(x) = \sum_{i=0}^{k-1} f_i x^i$ at the n distinct nonzero elements of $\mathbb{F}_q$. Given the message polynomial $f(x)$ and n distinct nonzero elements $\{\alpha_1, \alpha_2, \cdots, \alpha_n\}$, the RS codeword is generated by

$$\underline{c} = (f(\alpha_1), f(\alpha_2), \cdots, f(\alpha_n)).$$

Normally, the length $n = q-1$. The codeword is transmitted through a noisy channel and the received word is $\underline{y} = \underline{c} + \underline{e}$, where $\underline{e}$ is the error pattern.

Let $\mathbb{F}_q[x,y]$ denote the ring of bivariate polynomials defined over $\mathbb{F}_q$. A bivariate polynomial in $\mathbb{F}_q[x,y]$ is given as

$$Q(x,y) = \sum_{i,j} a_{i,j} x^i y^j, \tag{1}$$

where $a_{i,j} \in \mathbb{F}_q$. Then the $(1,k-1)$-weighted degree of $Q(x,y)$, denoted as $\deg_{1,k-1} Q(x,y)$, is the maximum over all $i+(k-1)j$ with $a_{i,j} \neq 0$. To arrange uniquely bivariate polynomials in a fixed order, it is necessary to extend the notion of $(1,k-1)$-weighted degree to a monomial order. Explicitly, the $(1,k-1)$-weighted degree lexicographic order [21] based on y is defined as $x^{i_1}y^{j_1} \prec x^{i_2}y^{j_2}$ if and only if

$$i_1 + (k-1)j_1 < i_2 + (k-1)j_2 \tag{2}$$

or

$$i_1 + (k-1)j_1 = i_2 + (k-1)j_2 \quad \text{and} \quad j_1 < j_2. \tag{3}$$

By a fixed monomial order $\prec$, a polynomial $Q(x,y) = \sum_{i,j} a_{i,j} x^i y^j$ can be rewritten uniquely as

$$Q(x,y) = \sum_{j=0}^{\gamma} a_j \phi_j(x,y) \tag{4}$$

with leading coefficient $a_\gamma \neq 0$. The integer γ is called the rank of $Q(x,y)$, written as $\mathrm{Rank}(Q)$. Thus each polynomial in $\mathbb{F}_q[x,y]$ has a well-defined leading term under the monomial order $\prec$. Denote $D_{r,s}Q(x,y)$ as the (r,s)-th Hasse derivative of $Q(x,y)$, which can be defined as

$$D_{r,s}Q(x,y) = \sum_{i,j} \binom{i}{r}\binom{j}{s} a_{i,j} x^{i-r} y^{j-s}. \tag{5}$$

It is necessary to indicate the relationship among the parameters of GS algorithm. Note that the factors $y - f(x)$ with $\deg f(x) \leq k-1$ are the focus of our interest. Then the weighted degree of y is naturally assigned to be $k-1$. Denote δ as the $(1,k-1)$ weighted-degree of the polynomial $Q(x,y)$. Note that the number of coefficients of $Q(x,y)$, $N_{1,k-1}(\delta)$, can be regarded as the number of monomials $x^i y^j$ with $i + (k-1)j \leq \delta$. Thus

$$\begin{aligned} N_{1,k-1}(\delta) &= \sum_{j=0}^{\lfloor \frac{\delta}{k-1} \rfloor} (\delta + 1 - j(k-1)) \\ &= (\lfloor \frac{\delta}{k-1} \rfloor + 1)(\delta + 1 - \frac{k-1}{2} \lfloor \frac{\delta}{k-1} \rfloor). \end{aligned} \tag{6}$$

As there are $\binom{m+1}{2}$ choices of (r,s) for $0 \leq r+s < m$, the interpolation problem is to solve a system of

$$C = \frac{nm(m+1)}{2} \tag{7}$$

homogeneous linear equations. Hence, we choose δ to be large enough such that

$$N_{1,k-1}(\delta) > C. \tag{8}$$

This ensures a nonzero solution to the interpolation problem.

On the other hand, the polynomial $Q(x,y)$ can be divisible by the factor $y - f(x)$ provided that $\delta \leq m(n-t) - 1$ [11]. Replacing δ with $m(n-t)-1$, we have

$$N_{1,k-1}(m(n-t)-1) > C. \tag{9}$$

Thus the error-correction capability t_m is defined as the maximum value for the multiplicity $m \geq 1$ satisfying [22]

$$t_m = \max\{t : N_{1,k-1}(m(n-t)-1) > C\}. \tag{10}$$

The list-size is defined as $L_m = \max\{L : \mathrm{Rank}(y^L) \leq C\}$, which is given by [22]

$$L_m = \lfloor \sqrt{(\frac{k+1}{2(k-1)})^2 + \frac{nm(m+1)}{k-1}} - \frac{k+1}{2(k-1)} \rfloor. \tag{11}$$

Hence, the error-correction capability t_m and the corresponding list-size L_m grow with multiplicity m as

$$t_{m_i} \leq t_{m_{i+1}} \quad \text{and} \quad L_{m_i} \leq L_{m_{i+1}}, \quad \text{if} \quad m_i < m_{i+1}. \tag{12}$$

The iterative polynomial construction algorithm [21] [9] has been widely recognized as the most efficient implementation algorithm for interpolation, which exhibits a complexity $O(n^2 m^4 L)$. It constructs a set of polynomials $\mathcal{G} = \{g_j(x,y) | 0 \leq j \leq L\}$ such that each polynomial $g_j(x,y)$ passes through all the points (x_i, y_i) at least m times. The minimal polynomial $Q(x,y)$ is given by the polynomial with the minimal rank among the output of the algorithm.

III. The Progressive Interpolation Approach

In this section, we present a progressive interpolation approach in order to keep the multiplicity as small as possible.

A. Outline of the Approach

Let $m_1, \cdots, m_\rho, \cdots, m_\tau$ be an increasing sequence of multiplicities. The progressive approach for GS decoding of RS codes is outlined as follows and also illustrated in Fig. 1, where ρ is the progressive index that is initialized as one.

The algorithm is implemented in an iterative manner, whose basic idea is that the current iteration can use the results calculated in the last iteration. At the ρ-th iteration, there are two phases: progressive interpolation and factorization. The progressive interpolation step is to find a set of polynomials passing through all points m_ρ times, which is denoted by

$$\mathcal{G}^{(\rho)} = \{g_0^{(\rho)}, g_1^{(\rho)}, \cdots, g_{L_\rho}^{(\rho)}\}. \tag{13}$$

This can be done by using Koetter's iterative interpolation algorithm [21] [9].

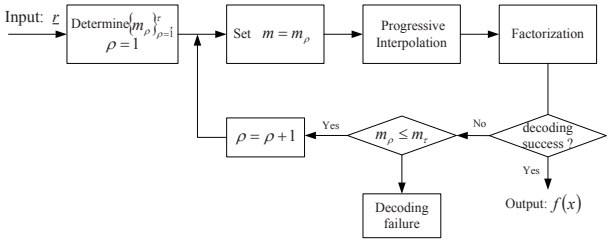

Fig. 1. The block diagram of the progressive interpolation.

Let $Q^{(\rho)}(x, y)$ be the polynomial with minimal rank selected from $\mathcal{G}^{(\rho)}$. The factorization is to find a list of message polynomial candidates satisfying

$$\mathcal{L} = \{f(x)|(y - f(x))|Q^{(\rho)}(x, y), \deg f(x) < k\}. \quad (14)$$

The factorization can be implemented by Roth-Ruckenstein algorithm [11]. If the output list $\mathcal{L}$ is not empty, terminate the algorithm by outputting the polynomial whose corresponding codeword has the minimal Hamming distance from the received word. Otherwise, increase the progressive index ρ by one and assign the multiplicity value as $m_{\rho+1}$ such that the error-correction capability improves to $t_{\rho+1} > t_\rho$.

B. The Progressive Interpolation

It is critical to observe that the minimal polynomial constructed by Koetter's interpolation algorithm is independent with the order of calculating Hasse derivative evaluation $\Delta_j = D_{r,s}g_j(x_i, y_i)$. At the beginning of each iteration, the polynomial set $\mathcal{G}^{(\rho)}$ is modified by [13]

$$\mathcal{G}^{(\rho)} = \{g_j|\mathrm{Rank}(g_j) \le C_\rho\}. \quad (15)$$

in order to eliminate those polynomials with leading order over $C_\rho = \frac{nm_\rho(m_\rho+1)}{2}$. Due to the rearrangement of the parameters (r, s) in calculating the Hasse derivative evaluation, we will show that the minimal polynomial $Q^{(\rho)}(x, y)$ can be constructed in a progressive way by using the interpolation results from the $(\rho - 1)$-th iteration.

For $1 \le \nu \le C_{\rho-1}$, the ν-th constraint is defined as $D_{r,s}g_j^{(\nu)}(x_i, y_i) = 0$ for $0 \le r + s < m_{\rho-1}$, where $D_{r,s}g_j^{(\nu)}(x_i, y_i)$ is the (r, s)-th Hasse derivative evaluation at the point (x_i, y_i) [21]. Define

$$\mathfrak{g}^{(\nu-1)} = \{g_j^{(\nu-1)}|D_{r,s}g_j^{(\nu)}(x_i, y_i) \ne 0\}. \quad (16)$$

Let the polynomial with the minimal rank in $\mathfrak{g}^{(\nu-1)}$ be

$$f^{(\nu-1)} = \min\{g_j^{(\nu-1)}|g_j^{(\nu-1)} \in \mathfrak{g}^{(\nu-1)}\}. \quad (17)$$

Denote the index and the Hasse derivative evaluation of $f^{(\nu-1)}$ as j^* and Δ^*, respectively. We only need to update the polynomials in $\mathfrak{g}^{(\nu-1)}$. The process of progressive interpolation can be illustrated as follows.

Step 1 Perform Koetter's interpolation algorithm for the constraints defined by $m_{\rho-1}$.
Initialization: $\mathcal{G}_0^{(\rho-1)} = \{1, y, \cdots, y^{L_{\rho-1}}\}$.
Iterations: For $1 \le \nu \le C_{\rho-1}$, the polynomials $g_j^{(\nu)}$ can be modified from $g_j^{(\nu-1)} \in \mathfrak{g}^{(\nu-1)}$ as

$$g_j^{(\nu)} = \begin{cases} \Delta^* g_j^{(\nu-1)} - \Delta_j f^{(\nu-1)}, & \text{if} \quad j \ne j^* \\ (x - x_i)f^{(\nu-1)}, & \text{if} \quad j = j^* \end{cases}.$$

Note that $\mathrm{Rank}(Q^{(\rho-1)}) < \mathrm{Rank}(y^{L_{\rho-1}+1})$. Thus the polynomial set $\mathcal{G}_\nu^{(\rho)}$ can be expanded from $\mathcal{G}_\nu^{(\rho-1)}$ by the following steps.

Step 2 Consider the $C_{\rho-1}$ constraints defined by $m_{\rho-1}$ at first. Let $\widehat{\mathcal{G}}_\nu^{(\rho)}$ be the polynomial set at the ν-th iteration with the progressive number ρ such that

$$\widehat{\mathcal{G}}_\nu^{(\rho)} = \mathcal{G}_\nu^{(\rho)} \bigcup \Delta\mathcal{G}_\nu^{(\rho)}, \quad (18)$$

where $\Delta\mathcal{G}_\nu^{(\rho)} = \{g_{L_{\rho-1}+1}^{(\nu)}, \cdots, g_{L_\rho}^{(\nu)}\}$. In other words, the polynomials in $\mathcal{G}_\nu^{(\rho)}$ do not need to be modified for all $1 \le \nu \le C_{\rho-1}$. Thus it needs to construct $\Delta\mathcal{G}_\nu^{(\rho)}$, which can be initialized as

$$\Delta\mathcal{G}_0^{(\rho)} = \{y^{L_{\rho-1}+1}, \cdots, y^{L_\rho}\}. \quad (19)$$

Note that $\mathrm{Rank}(Q^{(\rho-1)}) < \mathrm{Rank}(y^{L_{\rho-1}+1})$. Thus the polynomial $f^{(\nu)}$ can be shared between the $(\rho - 1)$-th and ρ-th progressive iteration for $1 \le \nu \le C_{\rho-1}$. Then the polynomials in $\Delta\mathcal{G}_\nu^{(\rho)}$ can be modified using the identified polynomial $f^{(\nu)}$ in $(\rho - 1)$-th progressive iteration.

Step 3 Notice that the polynomial set $\widehat{\mathcal{G}}^{(\rho)}$ needs to satisfy all C_ρ constraints. Now $\widehat{\mathcal{G}}^{(\rho)}$ has been updated for $C_{\rho-1}$ constraints. In the following, the polynomials in $\widehat{\mathcal{G}}^{(\rho)}$ can be modified such that $D_{r,s}g_j^{(\nu)}(x_i, y_i) = 0$ for $m_{\rho-1} \le r + s < m_\rho$ and $0 \le j \le L_\rho$.

Based on the above discussion of the progressive interpolation, we summarize the progressive process for GS decoding of RS codes in Algorithm 1.

IV. Performance Analysis

We will compare the interpolation using the progressive approach with the conventional interpolation using a prescribed multiplicity in terms of average computational complexity. The complexity is measured by the number of arithmetic calculations, i.e., additions and multiplications.

We first consider the computational complexity of the conventional interpolation over different choices for the multiplicity. Here the number of operations over $\mathbb{F}_q$ in the iterative interpolation algorithm is $O(C^2L)$, where $C = \frac{nm(m+1)}{2}$ is the total number of linear constraints. As shown in Table I, the decoding parameters and the computational complexity in the conventional interpolation are summarized for the $(63, 12)$ and $(63, 32)$ RS codes.

We have evaluated the performance of GS algorithm using the progressive approach to decode the $(63, 12)$ and $(63, 32)$ RS codes with binary phase-shift keying (BPSK) signalling over additive white Gaussian noise (AWGN) channels. The performance is measured by the frame-error-rate (FER)

Algorithm 1 The Progressive Interpolation Approach for GS algorithm

se

Input: Points $\{(x_i, y_i)\}_{i=0}^{n-1}$ and the multiplicity m_ρ and the list-size L_ρ.

Initialization: i. Set the progressive index $\rho = 1$;
ii. $\mathcal{G}_0 = \{1\}$, $m_0 = 0$ and $L_0 = 0$;

Iterations:

S1.(Progressive Interpolation)

Initialize $\widehat{\mathcal{G}}_\nu^{(\rho)} = \mathcal{G}_\nu^{(\rho)} \bigcup \{y^{L_{\rho-1}+1}, \cdots, y^{L_\rho}\}$;

for $(i = 0;\ i < n;\ i++)$

S1.1 for $(0 \le r + s < m_{\rho-1})$

Compute $\Delta_j = D_{r,s} g_j(x_i, y_i)$;

$g_j = \Delta^* g_j - \Delta_j f,\ L_{\rho-1} + 1 \le j \le L_\rho$;

S1.2 for $(m_{\rho-1} \le r + s < m_\rho)$

Compute $\Delta_j = D_{r,s} g_j(x_i, y_i),\ 0 \le j \le L_\rho$;

$j^* = \arg \min_{\Delta_j \neq 0} g_j(x, y), \Delta^* = \Delta_{j^*}, f = g_{j^*}$;

if $(j \neq j^*)$

Compute $g_j = \Delta^* g_j - \Delta_j f$;

else $g_{j^*} = (x - x_i) f$;

Find $Q_\rho(x, y) = \min_{0 \le j \le L_\rho} \{g_j | g_j \in \mathcal{G}^{(\rho)}\}$;

S2. (Factorization) Perform Roth-Ruckenstein algorithm to check if the output list $\mathcal{L}$ is nonempty. If $\mathcal{L} \neq \emptyset$, the decoder declares "*decoding success*" and output $f(x)$. Otherwise, go to S3.

S3. (Update) $\rho \leftarrow \rho + 1$ and update m_ρ. If $m_\rho > m_\tau$ and $\mathcal{L} = \emptyset$, the decoder declares "*decoding failure*." Otherwise, $\mathcal{G}_\nu^{(\rho)} \leftarrow \widehat{\mathcal{G}}_\nu^{(\rho-1)}$ and go to S1.

TABLE I
THE COMPUTATIONAL COMPLEXITY OF THE CONVENTIONAL INTERPOLATION WITH DIFFERENT MULTIPLICITIES

RS codes	Multiplicity	Constraints	Capability	List-Size	Complexity
$(63, 12)$	$m_1 = 1$	63	30	2	7.938×10^3
	$m_2 = 2$	189	33	5	1.7861×10^5
	$m_3 = 3$	378	34	7	1.0002×10^6
	$m_4 = 5$	945	35	12	1.0716×10^7
	$m_{GS} = 12$	4914	36	29	7.0027×10^8
$(63, 32)$	$m_1 = 1$	63	15	1	3.969×10^3
	$m_2 = 2$	189	16	3	1.0716×10^5
	$m_3 = 3$	378	17	4	5.7154×10^5
	$m_{GS} = 8$	2268	18	11	5.6582×10^7

v.s. SNR. The error-correction performance of decoding the $(63, 12)$ and $(63, 32)$ RS codes are shown in Fig. 2 and Fig. 3, respectively. For practical implementations, we set a threshold value m_τ in the interpolation using the progressive approach.

Fig. 4 shows that the complexity reduction of the interpolation using the progressive approach with $m_\tau = 5$ compared to the conventional interpolation with the multiplicity $m = 1, 3, 5$ for decoding the $(63, 12)$ RS code. From Fig. 4, we can see that there is a significant complexity reduction for the proposed interpolation algorithm compared with the conventional interpolation for the multiplicity $m = 5$. Moreover, we achieve a significant complexity reduction in the working region for

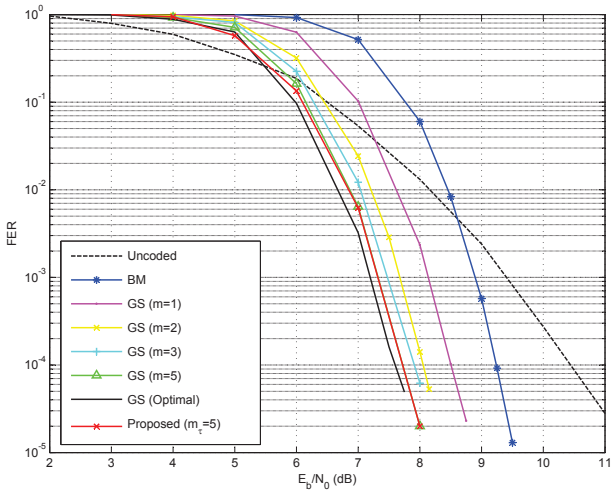

Fig. 2. The Performance of List Decoding Algorithms for RS (63, 12) codes.

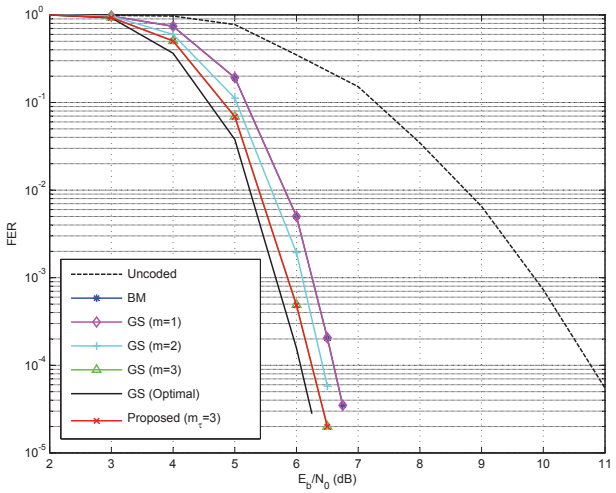

Fig. 3. The Performance of List Decoding Algorithms for RS (63, 32) codes.

SNR $\ge$ 7.0 dB since most of received words can be decoded successfully with $m = 1$. For example, at SNR = 8.0 dB, where the FER is about 10^{-5}, the complexity of the proposed interpolation using the progressive approach can be reduced by 99.97% compared with the iterative interpolation in GS algorithm. Due to the elimination of unnecessary polynomials [13], the complexity of the proposed algorithm is still less than GS algorithm with $m = 1$.

Fig. 5 shows that the complexity reduction of the interpolation using the progressive approach with $m_\tau = 3$ compared to the conventional interpolation with the multiplicity $m = 1, 2, 3$ for decoding the $(63, 32)$ RS code. From Fig. 5, we can see that there is also a significant complexity reduction in the working region for SNR $\ge$ 5.0 dB. For example, at SNR = 6.5 dB, where the FER is about 10^{-5}, the complexity of the proposed interpolation using the progressive approach can be reduced by 99.28% compared with the conventional interpolation in GS algorithm.

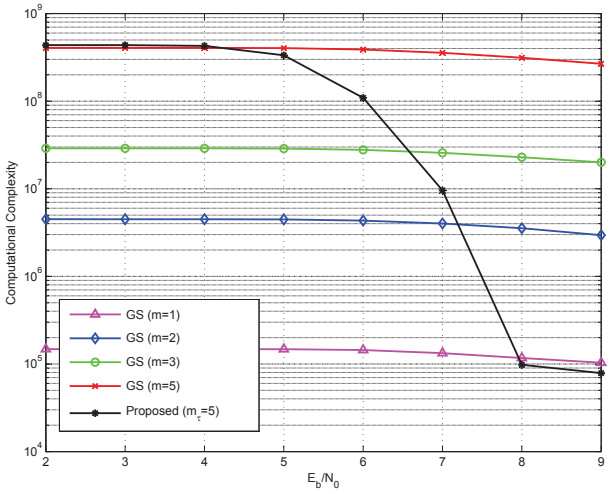

Fig. 4. The complexity reduction using the progressive approach in the interpolation for RS (63, 12) codes.

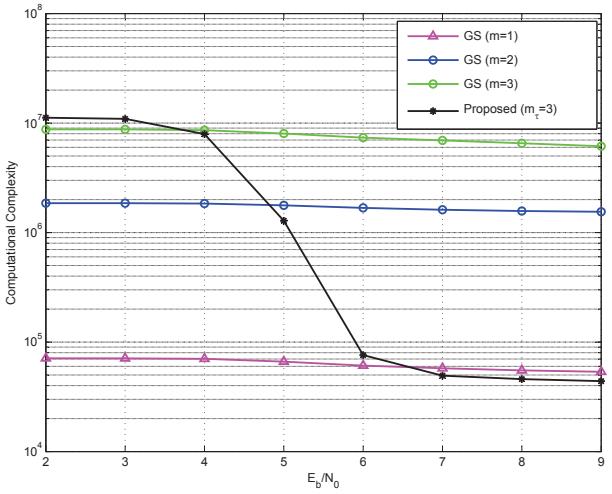

Fig. 5. The complexity reduction using the progressive approach in the interpolation for RS (63, 32) codes.

V. Conclusion

In this paper, a progressive approach to the interpolation has been proposed for GS decoding of RS codes. Since the most likely errors patterns can be corrected by GS list decoding algorithm with a small multiplicity value, the interpolation will be terminated by the progressive approach as soon as the decoder declares decoding success. An important advantage of the progressive approach is that the procedure will be modified on the basis of the previous interpolation results. It has been shown the progressive interpolation approach achieves a significant complexity reduction compared to the conventional interpolation approach.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (NSFC) under Grants 11126047, 61003227 and 61001094.

References

[1] E. R. Berlekamp, *Algebraic Coding Theory*. MacGraw-Hill (New York NY), 1968.

[2] J. L. Massey, "Shift-register synthesis and BCH decoding," *IEEE Trans. Inform. Theory*, vol. 15, no. 1, pp. 122–127, Jan. 1969.

[3] S. H. Y. Sugiyama, M. Kasahara and T. Namekawa, "A method for solving key equation for decoding Goppa codes," *Inf. Control*, vol. 27, pp. 87–99, 1975.

[4] L. R. Welch and E. R. Berlekamp, "Error correction for algebraic block codes," U.S.A., DEC. 1986.

[5] P. Elias, "List decoding for noisy channels," Res. Lab. Electron., MIT, Cambridge, MA, Tech. Rep., 1957.

[6] J. M. Wozencraft and R. S. Kennedy, "Modulattion and demodulation for probabilistic coding," *IEEE Transactions on Information Theory*, vol. IT-12, pp. 291–297, July 1966.

[7] Sudan, "Decoding of Reed Solomon codes beyond the error-correction bound," *J. Complexity*, vol. 13, pp. 180–193, 1997.

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

[9] R. Koetter, "Fast generalized minimun-distance decoding of algebraic geometry and Reed-Solomon codes," *IEEE Trans. Inform. Theory*, vol. 42, no. 3, pp. 721–737, 1996.

[10] R. Koetter and A. Vardy, "Algebraic soft-decision decoding of Reed-Solomon codes," *IEEE Trans. Inform. Theory*, vol. 49, no. 11, pp. 2809–2825, 2003.

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

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

[13] L. Chen, "Performance of Reed-Solomon codes using the Guruswami-Sudan algorithm with improved interpolation efficiency," *IET Commun.*, vol. 1, pp. 241–250, 2007.

[14] L. Chen, R. A. Carrasco, and M. Johnston, "Reduced complexity interpolation for list decoding Hermitian codes," *IEEE Trans. Wireless Commun.*, vol. 7, no. 11-1, 2008.

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

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

[17] K. Lee and M. E. O'Sullivan, "List decoding of Reed-Solomon codes from a Groebner basis perspective," pp. 645–658, 2008.

[18] P. Trifonov, "Efficient interpolation in the Guruswami-Sudan algorithm," *IEEE Trans. Inform. Theory*, vol. 56, no. 9, pp. 4341–4349, 2010.

[19] M. Ali and M. Kuijper, "A parametric approach to list decoding of Reed-Solomon codes using interpolation," *IEEE Trans. Inform. Theory*, vol. 57, no. 10, pp. 6718–6728, 2011.

[20] S. Tang, C. Li, and X. Ma, "Progressive list-enlarged algebraic soft decoding of Reed-Solomon codes," *IEEE Communication Letters*, vol. 16, no. 6, pp. 901–904, 2012.

[21] R. J. McEliece, "The Guruswami-Sudan decoding algorithm for Reed-Solomon codes," IPN Progress Report, Tech. Rep., 2003.

[22] T. K. Moon, *Error Correction Coding: Mathematical Methods and Algorithms*. Hoboken, New Jersey: John Wiley & Sons, Inc., 2005.