Progressive List-Enlarged Algebraic Soft Decoding of Reed-Solomon Codes

Siyun Tang, Li Chen, *Member, IEEE*, and Xiao Ma, *Member, IEEE*

***Abstract*—A progressive algebraic soft decoding algorithm is proposed for Reed-Solomon (RS) codes, aiming to reduce the computational complexity. The decoding starts with a small initial factorization output list size (OLS), then updates the OLS progressively leading to an incremental interpolation. Decoding will terminate either when the output contains a codeword that can be identified as the most likely one or the predefined maximal OLS is reached. The algorithm can adjust the decoding parameter according to the quality of the received information, optimizing its complexity to the minimal but necessary level.**

***Index Terms*—Algebraic soft decoding, complexity reduction, Koetter-Vardy algorithm, list decoding, progressive interpolation.**

I. Introduction

REED-SOLOMON (RS) codes are widely used nowadays. For an (n,k) RS code, where n and k are the length and dimension of the code respectively, the error-correction capability of the classical unique decoding algorithms [1] [2] is bounded by $\lfloor \frac{n-k}{2} \rfloor$. The algebraic list decoding algorithm [3] improves the error-correction bound to $n-\lfloor\sqrt{n(k-1)}\rfloor-1$. The algebraic soft decoding (ASD) algorithm [4] achieves further error-correction improvements. Due to its high decoding complexity, complexity reduction approaches including the facilitated reliability transform [5], the unnecessary interpolated polynomial elimination [6] and the re-encoding transformation [7] were introduced.

The ASD algorithm is flexible in nature since the decoding capability and complexity can be adjusted by varying its factorization output list size (OLS). In a good channel condition, most of the received words contain a small number of errors, and hence the ASD algorithm can perform most of its decodings with a small OLS. However, to decode a deeply corrupted received word which is atypical but may degrade the performance by orders of magnitude, a large OLS is required. Therefore, it is desirable to design an ASD algorithm that can adjust its decoding parameters to the quality of the received information. This inspires the proposed progressive list-enlarged algebraic soft decoding (PLEASD) algorithm.

II. Prerequisite knowledge

Let $\mathbb{F}_q=\{\alpha_0,\alpha_1,\cdots,\alpha_{q-1}\}$ denote the finite field of size q. Let x_i $(0\le i\le n-1)$ be n distinct nonzero elements of $\mathbb{F}_q$.

Manuscript received January 13, 2012. The associate editor coordinating the review of this letter and approving it for publication was M. Lentmaier.

The authors are with the School of Information Science and Technology, Sun Yat-sen University, Guangzhou 510006, China (e-mail: tangsiy@mail2.sysu.edu.cn, chenli55@mail.sysu.edu.cn, maxiao@mail.sysu.edu.cn).

This work is supported by the NSF of China and Guangdong with grants ID 61001094, 61172082 and 10451027501005078; by the Guangdong Provincial Program for High-Level Talents in University; and by the Young Academics Funding of Sun Yat-sen University with grant ID 101gpy29.

Digital Object Identifier 10.1109/LCOMM.2012.042512.112511

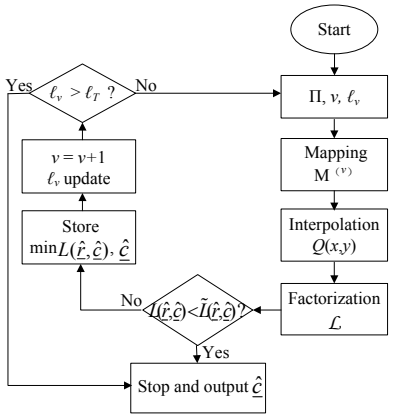

Fig. 1. The progressive list-enlarged algebraic soft decoding system.

Given a message polynomial $u(x)=u_0+u_1x+\cdots+u_{k-1}x^{k-1}$, the codeword of an (n,k) RS code can be generated as $\underline{c}=(u(x_0),u(x_1),\cdots,u(x_{n-1}))$. Assume $\underline{c}$ is transmitted over a memoryless channel and $\underline{r}=(r_0,r_1,\ldots,r_{n-1})$ is the received vector. Let Π denote the reliability matrix whose entries $\pi_{i,j}=\Pr\{c_j=\alpha_i|r_j\}$ for $0\le i\le q-1$ and $0\le j\le n-1$.

Matrix Π is mapped to a multiplicity matrix M. The ASD algorithm constructs a bivariate polynomial $Q(x,y)=\sum_{a,b\ge 0}Q_{ab}x^ay^b$ that has the minimal rank and satisfies all the interpolation constraints implied by matrix M. The monomials x^ay^b are ordered by the $(1,k-1)$-revlex order (ord) [8]. Let $x^{a'}y^{b'}$ be the leading monomial with $Q_{a'b'}\neq 0$, the $(1,k-1)$-weighted degree and rank of Q are defined as $\deg_{1,k-1}Q=a'+b'(k-1)$ and $\mathrm{rank}(Q)=\mathrm{ord}(x^{a'}y^{b'})$ respectively.

III. PLEASD Algorithm

A. System Model

The system model of the PLEASD algorithm is shown by Fig. 1, where v is the iteration index that is initialized as 1.

Given an increasing OLS sequence $\ell_1,\ldots,\ell_v,\ldots,\ell_T$, where ℓ_v denotes the decoding OLS of iteration v and ℓ_T is the predefined maximal OLS. To produce a series of multiplicity matrices $\mathrm{M}^{(1)},\ldots,\mathrm{M}^{(v)},\ldots,\mathrm{M}^{(T)}$, the PLEASD algorithm will perform Algorithm A of [4] iteratively. At each iteration, it is testified whether the following equation [9] holds:

$$\ell_v=\left\lfloor\frac{\Delta_{1,k-1}(\mathcal{C}(\mathrm{M}))}{k-1}\right\rfloor, \tag{1}$$

where $\mathcal{C}(\mathrm{M})=\sum_{i,j}m_{i,j}(m_{i,j}+1)/2$ is the cost of matrix M and $\Delta_{1,k-1}(\mathcal{C}(\mathrm{M}))=\deg_{1,k-1}x^ay^b$ with $\mathrm{ord}(x^ay^b)=\mathcal{C}(\mathrm{M})$. Once it is held, the matrix $\mathrm{M}^{(v)}$ that corresponds to the OLS of ℓ_v is produced. Let $m_{i,j}^{(v)}$ be the entry of $\mathrm{M}^{(v)}$. With the nature of the iterative reliability mapping $\Pi\mapsto\mathrm{M}$, we have $m_{i,j}^{(v)}\ge m_{i,j}^{(v-1)}$.

Given the multiplicity matrix $\mathrm{M}^{(v)}(1\le v\le T)$, the interpolation is to find a polynomial $Q_{\min}^{(v)}(x,y)$ that has the minimal rank and satisfies all the interpolation constraints

defined by $\mathrm{M}^{(v)}$. This is equivalent to solving a linear system derived from $\mathrm{M}^{(v)}$, and can be implemented efficiently by Koetter's algorithm [8]. It delivers a group of polynomials $\mathcal{G}=\{g_0,g_1,\cdots,g_{\ell_v}\}$ with an initial group $\mathcal{G}_0=\{1,y,\cdots,y^{\ell_v}\}$.

Let $Q_{\min}^{(v)}$ be the minimal polynomial of $\mathcal{G}$. Factorization [10] is to determine a list of decoding output candidates whose property can be characterized by

$$\mathcal{L}=\{f(x):\deg f(x)<k, y-f(x)|Q_{\min}^{(v)}\}. \tag{2}$$

If there exists $f(x)\in\mathcal{L}$ whose corresponding codeword can be identified (using the lemma shown in the next subsection) as the most likely codeword, the decoding will be terminated and $f(x)$ is given as the output. Otherwise, the iteration index will be updated as $v=v+1$, and it is followed by the OLS update. The decoding terminates once $\ell_v>\ell_T$. During the decoding, a possible codeword candidate $\underline{\hat{c}}$ is identified and stored in memory. If $\ell_v>\ell_T$ and the most likely codeword has not been found, the stored codeword $\underline{\hat{c}}$ is given as the output.

B. A Sufficient Condition for the Most Likely Codeword

Let $\underline{\hat{r}}=(\hat{r}_0,\hat{r}_1,\cdots,\hat{r}_{n-1})$ be the hard-decision received vector with $\hat{r_j}\in\mathbb{F}_q$. The maximum likelihood (ML) decoding is to find a codeword $\underline{\hat{c}}=(\hat{c}_0,\hat{c}_1,\ldots,\hat{c}_{n-1})$ that maximizes the metric $\sum_{0\le j\le n-1}\log(\pi_{i_j,j})$, where $i_j=\mathrm{index}\{i\mid\alpha_i=\hat{c}_j\}$. Let π_j^{1st} and π_j^{2nd} denote the largest and the second largest values in the j-th column of Π respectively, we have

$$\sum_{0\le j\le n-1}\log(\pi_{i_j,j})=\sum_{0\le j\le n-1}\log(\pi_j^{1st})-\sum_{j:\hat{c}_j\neq\hat{r}_j}(\log(\pi_j^{1st})-\log(\pi_{i_j,j})). \tag{3}$$

Therefore, the ML decoding is to find a codeword $\underline{\hat{c}}$ that minimizes the metric

$$L(\underline{\hat{r}},\underline{\hat{c}})=\sum_{j:\hat{c}_j\neq\hat{r}_j}(\log(\pi_j^{1st})-\log(\pi_{i_j,j})). \tag{4}$$

Reorder all the elements in $\{\log(\pi_j^{1st})-\log(\pi_j^{2nd}):\hat{c}_j=\hat{r}_j\}$ as $\log(\pi_{j_1}^{1st})-\log(\pi_{j_1}^{2nd})\le\log(\pi_{j_2}^{1st})-\log(\pi_{j_2}^{2nd})\le\cdots$. Define

$$\tilde{L}(\underline{\hat{r}},\underline{\hat{c}})=\sum_{t=1}^{d_{\min}-d}(\log(\pi_{j_t}^{1st})-\log(\pi_{j_t}^{2nd})), \tag{5}$$

where $d_{\min}=n-k+1$ is the minimum Hamming distance of the code and d is the Hamming distance between $\underline{\hat{c}}$ and $\underline{\hat{r}}$. The following lemma [11] can be used to identify the most likely transmitted codeword.

Lemma 1. If a codeword $\underline{\hat{c}}$ satisfies

$$L(\underline{\hat{r}},\underline{\hat{c}})\le\tilde{L}(\underline{\hat{r}},\underline{\hat{c}}), \tag{6}$$

then there is no codeword which is more likely than $\underline{\hat{c}}$.

It is worthwhile to point out that a codeword that satisfies the condition of (6) must be the most likely codeword. However, the most likely codeword may not satisfy the condition. To distinguish between these two cases, the former is referred to as the *verifiable* most likely codeword.

C. The Feasibility of the PLEASD Algorithm

It is important to note that the minimal bivariate polynomial found by Koetter's algorithm is independent of the interpolating schedule. For example, to find $Q(x,y)$ that passes through (x,α) and (y,β) with multiplicities of 2 and 1 respectively, one can first interpolate (x,α) and then (y,β) or vice versa. One can even first interpolate part of the constraints of (x,α) and (y,β), and then the remaining constraint of (x,α). Based on the argument, we will show that the computation of $Q_{\min}^{(v)}$ can be accomplished in a progressive way by using the intermediate results from the (v-1)-th iteration according to the following three steps.

Step 1: Perform Koetter's algorithm for constraints defined by $\mathrm{M}^{(v-1)}$. The intermediate results are stored as follows.

Initialization: $\mathcal{G}_0=\{1,y,\cdots,y^{\ell_{v-1}}\}$;

Iterations: For $1\le t\le\mathcal{C}(\mathrm{M}^{(v-1)})$, $\mathcal{D}_{p,q}g_h^{(t)}(x_j,\alpha_i)=0$ is the t-th constraint defined by $\mathrm{M}^{(v-1)}$, where $\mathcal{D}_{p,q}g_h^{(t)}(x_j,\alpha_i)$ is the (p,q)-th Hasse derivative evaluation at (x_j,α_i) and $0\le p+q<m_{i,j}^{(v-1)}$. Let $\Lambda^{(t-1)}=\{g_h^{(t-1)}|\mathcal{D}_{p,q}g_h^{(t-1)}(x_j,\alpha_i)\neq0\}$ and find

$$f^{(t-1)}=\min\{g_h^{(t-1)}\in\Lambda^{(t-1)}\}. \tag{7}$$

Then, we can update $\mathcal{G}_{t-1}$ to $\mathcal{G}_t=\{g_0^{(t)},g_1^{(t)},\cdots,g_{\ell_{v-1}}^{(t)}\}$ according to

$$g_h^{(t)}=\begin{cases}g_h^{(t-1)}, & g_h^{(t-1)}\notin\Lambda^{(t-1)}\\ [g_h^{(t-1)},f^{(t-1)}]_{\mathcal{D}}, & g_h^{(t-1)}\in\Lambda^{(t-1)}\ \&\ g_h^{(t-1)}\neq f^{(t-1)}\\ [xf^{(t-1)},f^{(t-1)}]_{\mathcal{D}}, & g_h^{(t-1)}=f^{(t-1)}\end{cases} \tag{8}$$

where $[g,f]_{\mathcal{D}}=\mathcal{D}_{p,q}f(x_j,\alpha_i)g-\mathcal{D}_{p,q}g(x_j,\alpha_i)f$ is the bilinear modification.

Note that for any $1\le t\le\mathcal{C}(\mathrm{M}^{(v-1)})$ and $h\le\ell_{v-1}$, the rank of $g_h^{(t)}$ is less than that of y^{ℓ_v}. We now extend the results to the solutions that correspond to iteration v by the horizontal expansion and vertical expansion successively.

Step 2 (Horizontal expansion): Firstly, consider the constraints defined by $\mathrm{M}^{(v-1)}$. Let $\tilde{\mathcal{G}}_t$ be the set of solutions at the t-th iteration and $1\le t\le\mathcal{C}(\mathrm{M}^{(v-1)})$. We will prove that $\tilde{\mathcal{G}}_t=\mathcal{G}_t\cup\{g_{\ell_{v-1}+1}^{(t)},\cdots,g_{\ell_v}^{(t)}\}$. That means, at the t-th iteration, we only need to find $\Delta\mathcal{G}_t=\{g_{\ell_{v-1}+1}^{(t)},\cdots,g_{\ell_v}^{(t)}\}$. This can be proved by the following induction.

Initially $\Delta\mathcal{G}_0=\{y^{\ell_{v-1}+1},\cdots,y^{\ell_v}\}$ and $\tilde{\mathcal{G}}_0=\mathcal{G}_0\cup\Delta\mathcal{G}_0$. Assume that at t-th iteration $\tilde{\mathcal{G}}_t=\mathcal{G}_t\cup\Delta\mathcal{G}_t$.

At the $(t+1)$-th iteration, let $\tilde{f}^{(t)}=\min\{g_h^{(t)}\in\tilde{\mathcal{G}}_t:\mathcal{D}_{p,q}g_h^{(t)}(x_j,\alpha_i)\neq0\}$. We must have either $\tilde{f}^{(t)}=f^{(t)}$ or $\tilde{f}^{(t)}\neq f^{(t)}$. For $\tilde{f}^{(t)}=f^{(t)}$, the first part of $\tilde{\mathcal{G}}_t$ is updated to $\mathcal{G}_{t+1}$ without modification. For $\tilde{f}^{(t)}\neq f^{(t)}$, no update is required for the first part of $\tilde{\mathcal{G}}_t$. In both cases, only the calculation for updating $\Delta\mathcal{G}_t$ is required. That is, $\tilde{\mathcal{G}}_{t+1}=\mathcal{G}_{t+1}\cup\Delta\mathcal{G}_{t+1}$.

Finally, $\tilde{\mathcal{G}}_S=\mathcal{G}_S\cup\Delta\mathcal{G}_S$, where $S=\mathcal{C}(\mathrm{M}^{(v-1)})$.

Step 3 (Vertical expansion): Notice that $\tilde{\mathcal{G}}_S$ satisfies the constraints defined by $\mathrm{M}^{(v-1)}$ but not the constraints defined by $\mathrm{M}^{(v)}$. We need to update $\tilde{\mathcal{G}}_S$ by taking the incremental constraints defined by $\mathrm{M}^{(v)}$ and $\mathrm{M}^{(v-1)}$ into account.

Let $\mathcal{G}$ be the output after the above three steps. It is possible to get different $\mathcal{G}$ for different interpolation schedules. However, the solution of the minimal polynomial $Q_{\min}^{(v)}$ is irrelevant to the schedule and its identity is unique.

The proposed algorithm is summarized as **Algorithm 1**.

IV. Performance and Complexity Comparisons

In this section, the PLEASD algorithm is compared with the conventional ASD algorithm by simulating the $(63,47)$ RS code over additive white Gaussian noise (AWGN) channel using binary-phase shift keying (BPSK) modulation. The performance is measured by the frame error rate (FER), while the computational complexity is measured by the average interpolation cost (over frames) $\mathcal{C}(\mathrm{M})$. The cost $\mathcal{C}(\mathrm{M})$ represents maximum number of iterations in the interpolation step

Algorithm 1 PLEASD for RS codes

Initialization: Initialize the OLS $\ell_0 = 0$ and the polynomial group $\mathcal{G}_0 = \{1\}$. Set $\mathrm{M}^{(0)} = [0]_{q\times n}$, $v = 1$ and $t = 0$.

Iterations:

1 ***Horizontal expansion*:** Initialize $\tilde{\mathcal{G}}_t = \mathcal{G}_t \cup \Delta\mathcal{G}_t$ with $\Delta\mathcal{G}_t = \{y^{\ell_{v-1}+1}, \cdots, y^{\ell_v}\}$. For $1 \le t \le \mathcal{C}(\mathrm{M}^{(v-1)})$, perform Koetter's algorithm for constraints defined by $\mathrm{M}^{(v-1)}$ to extend the set of solutions $\mathcal{G}_t$ to $\tilde{\mathcal{G}}_t = \mathcal{G}_t \cup \Delta\mathcal{G}_t$.

2 ***Vertical expansion*:** Find $\mathrm{M}^{(v)}$ and perform Koetter's algorithm for incremental constraints defined by $\mathrm{M}^{(v)}$ and $\mathrm{M}^{(v-1)}$ to get $\tilde{\mathcal{G}}_t$ for $\mathcal{C}(\mathrm{M}^{(v-1)})+1 \le t \le \mathcal{C}(\mathrm{M}^{(v)})$;

3 ***Factorization*:** Find $Q^{(v)}_{\min} = \min\tilde{\mathcal{G}}_S$ and $S = \mathcal{C}(\mathrm{M}^{(v)})$. Perform the factorization algorithm to check if the output list contains a verifiable most likely codeword. If so, report a decoding success and output $\underline{\hat{c}}$. Otherwise, store the minimal value of $L(\hat{\underline{r}}, \underline{\hat{c}})$ and $\underline{\hat{c}}$, then go to 4.

4 ***Updates*:** $v \leftarrow v + 1$ and update ℓ_v. If $\ell_v > \ell_T$, output $\underline{\hat{c}}$ and exit the decoding; otherwise, $\mathcal{G}_t \leftarrow \tilde{\mathcal{G}}_t$ and go to 1.

which dominates the decoding complexity. Please note that the average interpolation cost of the ASD algorithm is measured without incorporating other complexity reduction approaches, e.g., the re-encoding approach of [7].

A. The Conventional Parameters for the ASD Algorithm

In the ASD algorithm, the reliability transform (Algorithm of [4]) maps matrix Π to M in an iterative manner. In principle, any criterion to terminate the mapping process can be utilized to parameterize the ASD algorithm. This can be implemented by the following common ways. 1) For a given s, one can perform Algorithm A until the entries of M satisfy the condition $s = \sum_{i,j} m_{i,j}$ [4] [12]. 2) For a given cost $\mathcal{C}$, one can perform Algorithm A until $\mathcal{C}(\mathrm{M}) \ge \mathcal{C}$ [13]. 3) For any real value $\lambda > 0$, simply determine the multiplicity matrix by $\mathrm{M} = \lfloor \lambda\Pi \rfloor$ [5]. 4) For a given factorization OLS ℓ_v, one can perform Algorithm A until (1) is satisfied [9].

It is possible that to achieve a similar error-correction performance, running the ASD algorithm with different parameters may incur different decoding complexities. To verify this, Fig.2 compares the performance of ASD algorithm with parameters s and ℓ_T. It can be seen that performance can be improved by increasing the parameters s or ℓ_T. The FER performance with $s = 150, 250$ and 400 are almost the same as those with $\ell_T = 2, 4$ and 7 respectively. The corresponding average interpolation cost for different parameters are shown in Fig. 3. It shows the computational complexity increases by increasing the decoding parameters. To achieve the same error-correction performance, the ASD algorithm with parameter s yields a slightly lower cost. However, in both cases, the complexity is almost insensitive to the channel signal-to-noise ratio (SNR), except the case with $s = 400$ which results in a slightly increased computational complexity by increasing the SNR. This is because in the high SNR region, the multiplicity values are only assigned for a limited number of entries in M. With a fixed value of s, such a multiplicity distribution results in a higher cost $\mathcal{C}(\mathrm{M})$.

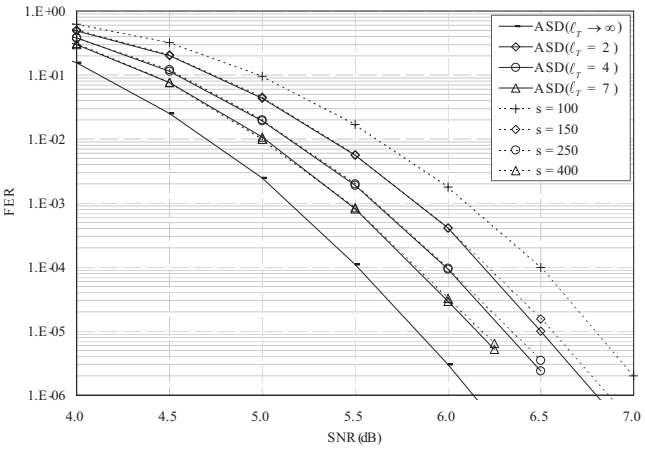

Fig. 2. Performance comparison of ASD algorithm for the $(63, 47)$ RS code with parameters ℓ_T and s.

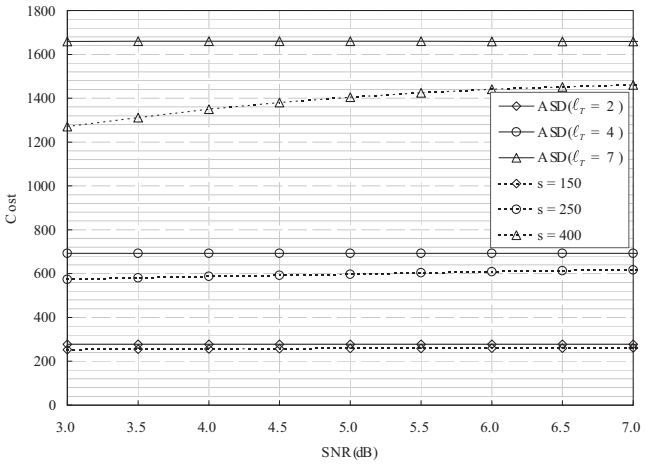

Fig. 3. Complexity comparison of ASD algorithm for the $(63, 47)$ RS code with parameters ℓ_T and s.

It is known that enhancing the factorization OLS results in an improved error-correction performance since more decoding output candidates are provided. More importantly, the OLS is a necessary parameter to be known in the decoding because OLS + 1 is the number of polynomials that participate into the interpolation process. Therefore, the factorization OLS is chosen as the progressive decoding parameter in this paper.

B. Comparison Between the PLEASD and ASD Algorithms

Performance Comparison The PLEASD algorithm is compared to the ASD algorithm that performs decoding with a fixed OLS of ℓ_T. The performances of the optimal ASD algorithm with $\ell_T \to \infty$ and the Welch-Berlekamp (WB) algorithm are shown as comparison benchmarks. Fig. 4 shows that the PLEASD algorithm can slightly improve the FER performance. This is due to the proposed algorithm generates the output list $\mathcal{L}$ by factorizing a sequence of bivariate polynomials $Q^{(1)}_{\min}, Q^{(2)}_{\min}, \cdots, Q^{(v)}_{\min}$. In contrast, the conventional ASD algorithm generates the output list $\mathcal{L}'$ by only factorizing one bivariate polynomial $Q^{(T)}_{\min}$. Since $\mathcal{L}' \subseteq \mathcal{L}$, it can be seen that even if the ASD algorithm fails to find the transmitted message polynomial, the PLEASD algorithm may still be able to find it. However, this situation rarely happens and hence only results in a marginal error-correction performance improvement.

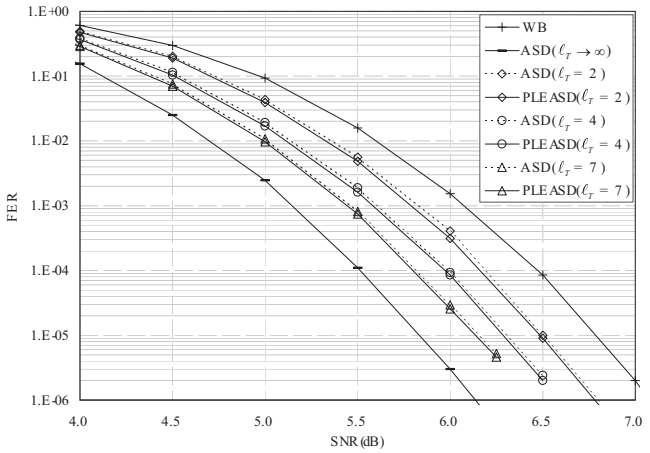

Fig. 4. Performance of the PLEASD algorithm for the $(63, 47)$ RS code.

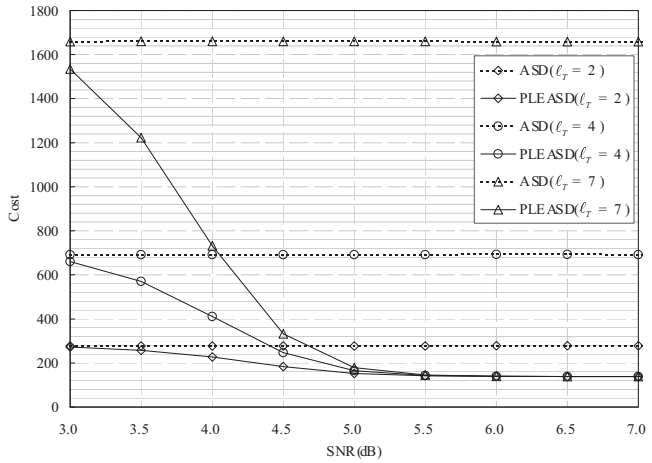

Fig. 5. Complexity of the PLEASD algorithm for the $(63, 47)$ RS code.

Complexity Comparison To implement the PLEASD algorithm, extra memory is needed to store the intermediate polynomials $f^{(t)}$. The extra memory requirement is upper bounded by $O(\mathcal{C}^2(\mathrm{M}^{(T)}))$. This is because we need to store at most $\mathcal{C}(\mathrm{M}^{(T)})$ polynomials $f^{(t)}$ and each of them has at most $\mathcal{C}(\mathrm{M}^{(T)})$ nonzero coefficients. Such a memory requirement is similar to the complexity reduction approach of [14]. It can be noticed that the PLEASD algorithm performs multiple factorizations followed by the output verifications. However, the complexity of multiple factorizations is still marginal and can be easily compensated by reducing the interpolation complexity. As for the output verification, the computational complexity in the real number field is upper bounded by $O(nT\ell_T)$. This is because there are at most $\sum_{v \le T} l_v$ different outputs. Each verification has a time complexity of $O(n)$.

Fig. 5 shows the average interpolation cost for the $(63, 47)$ RS code. It can be observed that in the medium to high SNR region (≥ 4 dB), significant complexity reduction can be achieved by using the PLEASD algorithm. The complexity reduction is mainly caused by the fact that the decoding may deliver the most likely codeword earlier (albeit with low probability at low SNR). More importantly, in the practically working region (SNR ≥ 4.5 dB), most of the received word can be decoded successfully with an OLS of 1. It results in a significant complexity reduction. Recall the results shown by Fig.4, with the same OLS of ℓ_T, both of the algorithms achieve a similar error-correction performance. Therefore, the complexity reduction is achieved without penalizing the algorithm's error-correction capability. The PLEASD algorithm enables the original ASD decoding process to be more flexible.

V. Conclusions

A progressive list-enlarged algebraic soft decoding algorithm for RS codes has been proposed. The algorithm exploits the fact that the interpolation constraints can be categorized by the nominal factorization OLS. The validity of the algorithm has been proved by showing that the intermediate interpolation results of the $(v–1)$-th iteration can be expanded to those of the v-th iteration by the horizontal and vertical expansions successively. Simulation results show that the progressive algorithm can reduce decoding complexity significantly without performance loss compared to the conventional ASD algorithm. It should be pointed out that this is a general approach that could be combined with other complexity reduction methods to further facilitate the decoding process.

Acknowledgment

The authors are grateful to Dr. Jingwei Zhang for useful discussions. They thank the anonymous reviewers for their constructive comments that help improve the presentation of this paper.

References

[1] L. Welch and E. R. Berlekamp, "Error correction for algebraic block codes," in *Proc. 1983 IEEE Int. Symp. Inform. Theory*.

[2] X. Ma and X. M. Wang, "On the minimal interpolation problem and decoding RS codes," *IEEE Trans. Inf. Theory*, vol. 46, no. 4, pp. 1573–1580, July 2000.

[3] 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, Sep. 1999.

[4] R. Koetter and A. Vardy, "Algebraic soft-decision decoding of Reed Solomon codes," *IEEE Trans. Inf. Theory*, vol. 49, pp. 2809–2825, 2003.

[5] W. J. Gross, F. R. Kschischang, R. Koetter, and P. G. Gulak, "Applications of algebraic soft-decision decoding of Reed-Solomon codes," *IEEE Trans. Commun.*, vol. 54, no. 7, pp. 1224–1234, July 2006.

[6] L. Chen, R. A. Carrasco, and E. G. Chester, "Performance of Reed-Solomon codes using the Guruswami-Sudan algorithm with improved interpolation efficiency," *IET Proc. Commun.*, vol. 1, pp. 241–250, 2007.

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

[8] R. J. McEliece, "The Guruswmi-Sudan decoding algorithm for Reed-Solomon codes," IPN Progress Rep., Tech. Rep. 42-153, 2003.

[9] L. Chen, R. A. Carrasco, and M. Johnston, "Soft-decision list decoding of Hermitian codes," *IEEE Trans. Commun.*, vol. 57, no. 8, pp. 2169–2176, Aug. 2009.

[10] R. M. Roth and G. Ruckenstein, "Efficient decoding of Reed-Solomon codes beyond half the minimum distance," *IEEE Trans. Inf. Theory*, vol. 46, no. 1, pp. 246–256, Jan. 2000.

[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] H. Xia and J. R. Cruz, "Reduced-complexity implementation of algebraic soft-decision decoding of Reed-Solomon codes," in *Proc. 2004 ICASSP*, vol. 5, pp. V–33–6.

[13] M. El-Khamy and R. J. McEliece, "Iterative algebraic soft-decision list decoding of Reed-Solomon codes," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 3, pp. 481–490, Mar. 2006.

[14] Y. Cassuto and J. Bruck, "On the average complexity of Reed-Solomon algebraic list decoders," in *Proc. 2005 IEE/IEEE Int. Symp. on Communication Theory and Applications*.