An Algorithm for Simultaneous Partial Inverses

Jiun-Hung Yu and Hans-Andrea Loeliger
Department of Information Technology and Electrical Engineering
ETH Zurich, Switzerland
Email: {yu, loeliger}@isi.ee.ethz.ch

***Abstract*—The simultaneous partial-inverse problem is similar to the multi-sequence shift-register synthesis problem. The paper introduces the problem and proposes a new algorithm for its solution. An application to decoding interleaved Reed-Solomon codes, beyond half the minimum distance, is also demonstrated.**

I. Introduction

In this paper, we propose an algorithm that solves the following problem.

Simultaneous Partial-Inverse (SPI) Problem: For $i = 1, 2, \ldots, L$, let $b^{(i)}(x)$ and $m^{(i)}(x)$ be nonzero polynomials over some field F with $\deg b^{(i)}(x) < \deg m^{(i)}(x)$. The problem is to find a nonzero polynomial $\Lambda(x) \in F[x]$ of the smallest degree such that

$$\deg\Big(b^{(i)}(x)\Lambda(x) \bmod m^{(i)}(x)\Big) < \tau^{(i)} \tag{1}$$

for given $\tau^{(i)} \in \mathbb{Z}$ with $1 \le \tau^{(i)} \le \deg m^{(i)}(x)$. □

It is not hard to prove (see Section II) that this problem has always a unique solution, up to a scale factor.

In the special case where $L = 1$, the SPI problem reduces to the Partial-Inverse Problem of [1], which includes computing inverses in $F[x]/m(x)$, computing Padé approximants, and solving the key equation for decoding Reed-Solomon codes as special cases.

The SPI problem for $L > 1$ is similar to the multi-sequence shift-register synthesis (MSSRS) problem [2]–[5] and can be used for similar purposes (such as [6]–[8]). Indeed, inspired by [8]–[10], we demonstrate the application of the proposed SPI algorithm to decoding a scheme of interleaved Reed-Solomon codes beyond half the minimum distance. However, the SPI problem is not identical to the MSSRS problem; e.g., the SPI problem has always a unique solution, but neither the original LFSRS problem [11] nor the MSSRS problem have this property.

Also, the algorithm proposed in this paper looks very similar to the MSSRS algorithms in [2]–[5], but it is not identical to any of them. Moreover, the proof of the proposed algorithm is a nontrivial generalization of the proof in [1] and does not resemble any of the proofs of MSSRS algorithms.

The paper is structured as follows. The existence and uniqueness of the SPI problem are proved in Section II, together with a bound on the degree of the solution. The new algorithm is proposed in Section III. The application to decoding interleaved Reed-Solomon codes is described in Section IV. The proof of the algorithm is given in Section V. Section VI concludes the paper.

The coefficient of x^d of a polynomial $b(x) \in F[x]$ will be denoted by b_d, and the leading coefficient (i.e., the coefficient of $x^{\deg b(x)}$) of a nonzero polynomial $b(x)$ will be denoted by $\operatorname{lcf} b(x)$.

II. Existence, Uniqueness, and Degree Bound

It is obvious that the SPI problem has always a solution: if $\Lambda(x) = \operatorname{lcm}(m^{(1)}(x), \ldots, m^{(L)}(x))$, the least common multiple of all $m^{(i)}(x)$, then

$$b^{(i)}(x)\Lambda(x) \bmod m^{(i)}(x) = 0. \tag{2}$$

In general, however, the SPI problem is solved by a polynomial of much smaller degree than $\operatorname{lcm}(m^{(1)}(x), \ldots, m^{(L)}(x))$, cf. Proposition 2 below.

Proposition 1 (Uniqueness). The solution $\Lambda(x)$ of the Simultaneous Partial-Inverse Problem is unique up to a scale factor. □

Proof: Let $\Lambda'(x)$ and $\Lambda''(x)$ be two solutions of the problem, which implies $\deg \Lambda'(x) = \deg \Lambda''(x) \ge 0$. Define

$$r'^{(i)}(x) \triangleq b^{(i)}(x)\Lambda'(x) \bmod m^{(i)}(x) \tag{3}$$

$$r''^{(i)}(x) \triangleq b^{(i)}(x)\Lambda''(x) \bmod m^{(i)}(x) \tag{4}$$

and consider

$$\Lambda(x) \triangleq \Big(\operatorname{lcf}\Lambda''(x)\Big)\Lambda'(x) - \Big(\operatorname{lcf}\Lambda'(x)\Big)\Lambda''(x). \tag{5}$$

Then

$$r^{(i)}(x) \triangleq b^{(i)}(x)\Lambda(x) \bmod m^{(i)}(x) \tag{6}$$

$$= \Big(\operatorname{lcf}\Lambda''(x)\Big)r'^{(i)}(x) - \Big(\operatorname{lcf}\Lambda'(x)\Big)r''^{(i)}(x) \tag{7}$$

by the natural ring homomorphism $F[x] \to F[x]/m^{(i)}(x)$. Clearly, (7) implies that $\Lambda(x)$ also satisfies (1) for every $1 \le i \le L$. But (5) implies $\deg \Lambda(x) < \deg \Lambda'(x)$, which is a contradiction unless $\Lambda(x) = 0$. Thus $\Lambda(x) = 0$, which means that $\Lambda'(x)$ equals $\Lambda''(x)$ up to a scale factor. □

Proposition 2 (Degree Bound). The solution $\Lambda(x)$ of the Simultaneous Partial-Inverse Problem satisfies

$$\deg \Lambda(x) \le \sum_{i=1}^{L} \Big(\deg m^{(i)}(x) - \tau^{(i)}\Big). \tag{8}$$

□

Proof: Let $\nu_i \triangleq \deg m^{(i)}(x) - \tau^{(i)}$ and $\nu \triangleq \sum_{i=1}^{L} \nu_i$. For $i = 1, \ldots, L$, consider the linear mappings

$$\varphi_i : F^{\nu+1} \to F^{\nu_i} \tag{9}$$

given by

$$(\Lambda_0, \ldots, \Lambda_\nu) \mapsto \Lambda(x) \triangleq \Lambda_0 + \Lambda_1 x + \ldots + \Lambda_\nu x^\nu \tag{10}$$

$$\mapsto r^{(i)}(x) \triangleq b^{(i)}(x)\Lambda(x) \bmod m^{(i)}(x) \tag{11}$$

$$\mapsto (r_0^{(i)}, \ldots, r_{\deg m^{(i)}(x)-1}^{(i)}) \tag{12}$$

$$\mapsto (r_{\tau^{(i)}}^{(i)}, \ldots, r_{\deg m^{(i)}(x)-1}^{(i)}). \tag{13}$$

Note that a polynomial $\Lambda_0 + \Lambda_1 x + \ldots + \Lambda_\nu x^\nu$ satisfies (1) if and only if $(\Lambda_0, \ldots, \Lambda_\nu) \in \ker \varphi_i$. But

$$\dim \left(\bigcap_{i=1}^{L} \ker \varphi_i \right) \geq \nu + 1 - \sum_{i=1}^{L} \nu_i \tag{14}$$

$$= 1 \tag{15}$$

and $\left(\bigcap_{i=1}^{L} \ker \varphi_i \right)$ is not trivial. There thus exists some nonzero polynomial $\Lambda_0 + \Lambda_1 x + \ldots + \Lambda_\nu x^\nu$ that satisfies (1) simultaneously for $i = 1, \ldots, L$. □

III. The Proposed Algorithm

The SPI problem as stated in Section I can be solved by Algorithm 1 on this page. Lines 1–8 are for initialization; the nontrivial part begins with line 9. Note that lines 21–23 simply swap $\Lambda(x)$ with $\Lambda^{(i)}(x)$, d with $d^{(i)}$, and κ with $\kappa^{(i)}$. The only actual computations are in lines 18 and 26. Note that in line 18, we have $\kappa = 0$ if $d \geq \deg m^{(i)}(x)$.

A. Some Explanations

We now begin to explain the algorithm (but the detailed proof of correctness will be given in Section V). To this end, we define the following quantities. For any nonzero $\Lambda(x)$ and any $i \in \{1, 2, \ldots, L\}$, let

$$\mathrm{rd}^{(i)}(\Lambda) \triangleq \deg\big(b^{(i)}(x)\Lambda(x) \bmod m^{(i)}(x)\big), \tag{16}$$

$$\delta_{\max}(\Lambda) \triangleq \max_{i \in \{1,\ldots,L\}} \Big(\mathrm{rd}^{(i)}(\Lambda) - \tau^{(i)} \Big), \tag{17}$$

and

$$i_{\max}(\Lambda) \triangleq \max \operatorname*{argmax}_{i \in \{1,\ldots,L\}} \Big(\mathrm{rd}^{(i)}(\Lambda) - \tau^{(i)} \Big), \tag{18}$$

the largest among the indices i that maximize $\mathrm{rd}^{(i)}(\Lambda) - \tau^{(i)}$, cf. Figure 1.

At any given time, the algorithm works on the polynomial $\Lambda(x)$. The inner **repeat** loop (lines 10–19) computes the quantities defined in (16)–(18): between lines 19 and 20, we have

$$i = i_{\max}(\Lambda),\ \delta = \delta_{\max}(\Lambda),\ d = \mathrm{rd}^{(i)}(\Lambda), \tag{19}$$

and also

$$\kappa = \mathrm{lcf}\big(b^{(i)}(x)\Lambda(x) \bmod m^{(i)}(x)\big). \tag{20}$$

In particular, the very first execution of the **repeat** loop (with $\Lambda(x) = 1$) yields

$$i = \max \operatorname*{argmax}_{i \in \{1,\ldots,L\}} \Big(\deg b^{(i)} - \tau^{(i)} \Big), \tag{21}$$

Algorithm 1

Simultaneous Partial-Inverse (SPI) Algorithm

Input: $m^{(i)}(x)$, $b^{(i)}(x), \tau^{(i)}$ for $i = 1, \ldots, L$.
Output: $\Lambda(x)$ as in the problem statement.

```
1   for i = 1, ..., L begin
2       Λ^(i)(x) := 0
3       d^(i) := deg m^(i)(x)
4       κ^(i) := lcf m^(i)(x)
5   end
6   Λ(x) := 1
7   δ := max_{i∈{1,...,L}} ( deg m^(i)(x) − τ^(i) )
8   i := 1
9   loop begin
10      repeat
11          if i > 1 begin i := i − 1 end
12          else begin
13              if δ ≤ 0 return Λ(x)
14              i := L
15              δ := δ − 1
16          end
17          d := δ + τ^(i)
18          κ := coefficient of x^d in
                    b^(i)(x)Λ(x) mod m^(i)(x)
19      until κ ≠ 0
        ---------------------------------------
20      if d < d^(i) begin
21          swap (Λ(x), Λ^(i)(x))
22          swap (d, d^(i))
23          swap (κ, κ^(i))
24          δ := d − τ^(i)
25      end
        ---------------------------------------
26      Λ(x) := κ^(i)Λ(x) − κ x^{d−d^(i)} Λ^(i)(x)
27  end
```

$d = \deg b^{(i)}(x)$, and $\kappa = \mathrm{lcf}\, b^{(i)}(x)$ between lines 19 and 20.

In the special case $L = 1$, lines 11–17 (excluding line 13) amount to $d := d - 1$; in this case, the algorithm reduces to the partial-inverse algorithm of [1].

The only exit from the algorithm is line 13. Since $\delta \geq \delta_{\max}(\Lambda)$, the condition $\delta \leq 0$ guarantees that $\Lambda(x)$ satisfies (1).

The algorithm maintains the auxiliary polynomials $\Lambda^{(i)}(x)$, $i = 1, \ldots, L$, which are all initialized to $\Lambda^{(i)}(x) = 0$. Thereafter, however, $\Lambda^{(i)}(x)$ become nonzero (after their first respective execution of lines 21–23) and satisfy

$$i_{\max}(\Lambda^{(i)}) = i. \tag{22}$$

The heart of the algorithm is line 26, which cancels the leading term in

$$b^{(i)}(x)\Lambda(x) \bmod m^{(i)}(x) \tag{23}$$

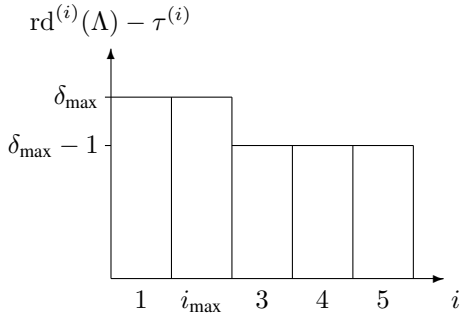

Fig. 1. Illustration of (17) and (18) for $i_{\max} = 2$.

(except for the first execution for each index i, see below). Line 26 is explained by the following lemma.

Lemma 1 (Remainder Decreasing Lemma). Let $\Lambda'(x)$ and $\Lambda''(x)$ be nonzero polynomials such that $i \triangleq i_{\max}(\Lambda') = i_{\max}(\Lambda'')$ and $\mathrm{rd}^{(i)}(\Lambda') \geq \mathrm{rd}^{(i)}(\Lambda'')$. Then $\delta_{\max}(\Lambda') \geq \delta_{\max}(\Lambda'')$ and the polynomial

$$\Lambda(x) \triangleq \kappa''\Lambda'(x) - \kappa' x^{d'-d''}\Lambda''(x) \tag{24}$$

with $d' \triangleq \mathrm{rd}^{(i)}(\Lambda')$, $\kappa' \triangleq \mathrm{lcf}(b^{(i)}(x)\Lambda'(x) \bmod m^{(i)}(x))$, $d'' \triangleq \mathrm{rd}^{(i)}(\Lambda'')$, and $\kappa'' \triangleq \mathrm{lcf}(b^{(i)}(x)\Lambda''(x) \bmod m^{(i)}(x))$ satisfies both

$$\mathrm{rd}^{(i)}(\Lambda) < \mathrm{rd}^{(i)}(\Lambda') \tag{25}$$

and

$$\delta_{\max}(\Lambda) \leq \delta_{\max}(\Lambda') \tag{26}$$

and either

$$i_{\max}(\Lambda) < i_{\max}(\Lambda'), \tag{27}$$

or

$$\delta_{\max}(\Lambda) < \delta_{\max}(\Lambda'). \tag{28}$$

□

The lemma is proved in Section V-A. It follows from (25)–(28) that the algorithm makes progress and eventually terminates.

For each index $i \in \{1, \ldots, L\}$, when line 26 is executed for the very first time, it is necessarily preceded by the swap in lines 21–23. In this case, line 26 reduces to

$$\Lambda(x) := -\Big(\mathrm{lcf}\, m^{(i)}(x)\Big) x^{\deg m^{(i)}(x) - \mathrm{rd}^{(i)}(\Lambda')}\Lambda'(x) \tag{29}$$

where $\Lambda'(x)$ is the value of $\Lambda(x)$ before the swap. It follows, in particular, that $\deg \Lambda(x) > \deg \Lambda'(x)$.

In any case, we always have

$$\deg(b^{(i)}(x)\Lambda(x) \bmod m^{(i)}(x)) < d \tag{30}$$

after executing line 26.

Finally, we note that every execution of the swap in lines 21–23 strictly reduces $d^{(i)}$. We also note that the execution of line 24 results in

$$\delta = \begin{cases} \delta_{\max}(\Lambda), & \text{if } \Lambda(x) \neq 0 \\ \deg m^{(i)} - \tau^{(i)}, & \text{if } \Lambda(x) = 0, \end{cases} \tag{31}$$

where the second case happens only once—the very first time—for each index $i \in \{1, \ldots, L\}$.

B. Complexity

The complexity of every iteration of lines 9–27 is dominated by the complexity of the inner **repeat** loop. Let $M(n)$ denote the complexity of the inner loop, where $n \triangleq \max \deg m^{(i)}(x)$. Due to (25)–(28) (Lemma 1), the algorithm executes at most $O(Ln)$ iterations of the outer loop. It follows that the overall complexity of the algorithm is $O(LnM(n))$.

As for the complexity of line 18, we first note that

$$\deg\big(b^{(i)}(x)\Lambda(x) \bmod m^{(i)}(x)\big) \leq d \tag{32}$$

is guaranteed before every execution of line 18.

In the special case where $m^{(i)}(x) = x^{\deg m^{(i)}(x)}$, line 18 amounts to

41 $\qquad \kappa := b_d^{(i)}\Lambda_0 + b_{d-1}^{(i)}\Lambda_1 + \ldots + b_{d-\nu}^{(i)}\Lambda_\nu$

where $\nu \triangleq \deg \Lambda(x)$ and where $b_\mu^{(i)} \triangleq 0$ for $\mu < 0$. In another special case where $m^{(i)}(x) = x^{\deg m^{(i)}(x)} - 1$ for all i, line 18 becomes

51 $\qquad \kappa := b_d^{(i)}\Lambda_0 + b_{[d-1]}^{(i)}\Lambda_1 + \ldots + b_{[d-\nu]}^{(i)}\Lambda_\nu$

with $b_{[\mu]}^{(i)} \triangleq b_{\mu \bmod n}^{(i)}$. In both cases, the computation of line 18 only requires $O(n)$ operations; the algorithm then has the complexity $O(Ln^2)$, which agrees with the complexity of the MSSRS algorithms in [3], [4].

IV. Application to Decoding Interleaved Reed-Solomon Codes

In this section, we briefly demonstrate the application of the proposed SPI algorithm to decoding interleaved Reed-Solomon codes beyond half the minimum distance.

A. The Codes

Let F be a finite field, let $\beta_0, \ldots, \beta_{n-1}$ be n different elements of F, let $m(x) \triangleq \prod_{\ell=0}^{n-1}(x - \beta_\ell)$, let $F[x]/m(x)$ be the ring of polynomials modulo $m(x)$, and let ψ be the evaluation mapping

$$\psi : F[x]/m(x) \to F^n : a(x) \mapsto \big(a(\beta_0), \ldots, a(\beta_{n-1})\big), \tag{33}$$

which is a ring isomorphism. Note that $\deg m(x) = n$.

We then define an $[n, k]$ Reed-Solomon code $\mathcal{C}$ with blocklength n and dimension k as the set

$$\mathcal{C} \triangleq \{c \in F^n : \deg \psi^{-1}(c) < k\}, \tag{34}$$

and consider an interleaved Reed-Solomon (IRS) code $\mathcal{C}_{\mathrm{IRS}}$ of depth L as a set such that every element of $\mathcal{C}_{\mathrm{IRS}}$ is a $L \times n$ matrix where every row corresponds to a codeword taken from $\mathcal{C}$.

B. Channel Model and Error Locator Polynomial

Now, let $A \in \mathcal{C}_{\mathrm{IRS}}$ denote a $L \times n$ matrix transmitted through a channel, and let

$$Y = A + E \tag{35}$$

be the received matrix, where the matrix $E \in F^{L \times n}$ represents the error that corrupts A. Moreover, let $a^{(i)}$ denote

the i-th row of A, $y^{(i)}$ the i-th row of Y, and $e^{(i)}$ the i-th row of E. We then have

$$y^{(i)} = a^{(i)} + e^{(i)}, \tag{36}$$

for $i = 1, \ldots, L$ and therefore

$$Y^{(i)}(x) = C^{(i)}(x) + E^{(i)}(x) \tag{37}$$

where $Y^{(i)}(x) \triangleq \psi^{-1}(y^{(i)})$, $C^{(i)}(x) \triangleq \psi^{-1}(a^{(i)})$, and $E^{(i)}(x) \triangleq \psi^{-1}(e^{(i)})$. Note that $\deg C^{(i)}(x) < k$ and $\deg E^{(i)}(x) < \deg m(x) = n$.

Clearly, every $C^{(i)}(x)$ can be decoded independently from the respective $Y^{(i)}(x)$ by the Berlekamp-Massey or by gcd-based decoding algorithms (or by the algorithm of [1]) provided that the number of erroneous symbols in the i-th row is not larger than $(n-k)/2$.

However, if the errors tend to be concentrated in columns of E, we may prefer to consider column errors rather than symbol errors. Indeed, it has been shown in [6], [8]–[10] that more than $(n-k)/2$ column errors can be corrected if $L > 1$.

Let $U \subset \{1, \ldots, n\}$ be the set indexing the nonzero columns of E. We then define, for any E, the error-locator polynomial

$$\Lambda_e(x) \triangleq \prod_{j \in U} (x - \beta_j). \tag{38}$$

It is easy to see that $\Lambda_e(x)$ satisfies

$$E^{(i)}(x)\Lambda_e(x) \bmod m(x) = 0 \tag{39}$$

for every $i \in \{1, \ldots, L\}$. In consequence, we have

$$\deg\big(Y^{(i)}(x)\Lambda_e(x) \bmod m(x)\big) < k + \deg \Lambda_e(x) \tag{40}$$

for every i.

C. Key Theorem

It has been shown in [8]–[10] that the rank of the matrix E can be relevant for decoding $\mathcal{C}_{\text{IRS}}$. In particular, every error pattern E with less than $n-k$ column errors can be corrected if the rank of E equals the number $|U|$ of column errors.

Let r denote the rank of the submatrix formed by the nonzero columns of E, and note that $r \le |U| = \deg \Lambda_e(x)$. We then have the following Theorem, which is similar to (but not identical with) Lemma 3 of [8] and generalizes Theorem 1 of [1].

Theorem 1. If

$$2|U| \le n - k + r - 1, \tag{41}$$

then the error locator polynomial (38) satisfies

$$\deg\Big(Y^{(i)}(x)\Lambda_e(x) \bmod m(x)\Big) < \frac{n+k+r-1}{2} \tag{42}$$

for all $i \in \{1, \ldots, L\}$. Conversely, for any Y and any $E \in F^{L \times n}$ (of rank r) and $t \in \mathbb{R}$ with

$$|U| \le t \le \frac{n-k+r-1}{2} \tag{43}$$

if some nonzero $\Lambda(x) \in F[x]$ with $\deg \Lambda(x) \le t$ satisfies

$$\deg\Big(Y^{(i)}(x)\Lambda(x) \bmod m(x)\Big) < n - t + r - 1 \tag{44}$$

for all $i \in \{1, \ldots, L\}$, then $\Lambda(x)$ is a multiple of $\Lambda_e(x)$. □

The proof of the direct part of Theorem 1 is easy: from (40), we have

$$\deg\Big(Y^{(i)}(x)\Lambda_e(x) \bmod m(x)\Big) < k + |U|, \tag{45}$$

and (41) implies $k + |U| \le \frac{n+k+r-1}{2}$. The (much longer) proof of the converse part is omitted due to the space constraints of this paper.

In the special case where $L = 1$, we have $r = 1$ (if $|U| > 0$) and the theorem reduces to Theorem 1 of [1].

In another special case where $r = |U|$, (41) reduces to $|U| < n - k$; in this case, $t = r$ satisfies (43), and (44) becomes

$$\deg\Big(Y^{(i)}(x)\Lambda(x) \bmod m(x)\Big) < n - 1. \tag{46}$$

D. Finding the Error Locator With the SPI Algorithm

Condition (40) and Theorem 1 suggest the following decoding algorithm.

Error Locating Algorithm:

1) Set $b^{(i)} = Y^{(i)}(x)$, $m^{(i)} = m(x)$, and $\tau^{(i)} = n - 1$ for every $i \in \{1, \ldots, L\}$.
2) Run the **SPI** algorithm.
3) If the returned polynomial $\Lambda(x)$ satisfies the condition
$$\deg\big(Y^{(i)}(x)\Lambda(x) \bmod m(x)\big) < k + \deg \Lambda(x) \tag{47}$$
for every $i \in \{1, \ldots, L\}$, then stop.
4) Otherwise, decrease all $\tau^{(i)}$ by 1 and continue the **SPI** algorithm.
5) Go to 3). □

Theorem 1 guarantees that this algorithm finds $\Lambda(x) = \gamma\Lambda_e(x)$ (for some nonzero $\gamma \in F$) provided that (41) is satisfied. This guarantee agrees with the guarantee in [8].

In the special case where $r = |U|$, it follows from (46) that the algorithm stops at the earliest possible moment (when (47) is checked for the first time); this special case is very likely if $L \ge n - k$.

Finally, it can be shown that coefficients $Y_\ell^{(i)}$ of $Y^{(i)}(x) = \sum_\ell Y_\ell^{(i)} x^\ell$ with $\ell < k$ are irrelevant for finding $\Lambda(x) = \gamma\Lambda_e(x)$ and can be set to zero. The remaining coefficients $Y_\ell^{(i)}$ are syndromes since $C_\ell^{(i)} = 0$ and $Y_\ell^{(i)} = E_\ell^{(i)}$ for $\ell \ge k$.

E. Finish Decoding

Having found an estimate of the error locator polynomial, decoding can be completed in any standard way [12] or by

$$C^{(i)}(x) = \frac{Y^{(i)}(x)\Lambda(x) \bmod m(x)}{\Lambda(x)} \tag{48}$$

(as proposed in [13]) or by means of the following proposition:

Proposition 3. If $\Lambda(x) = \gamma\Lambda_e(x)$ (for some nonzero $\gamma \in F$) satisfies $\deg \Lambda(x) \le n - k$, then

$$C^{(i)}(x) = Y^{(i)}(x) \bmod \tilde{m}(x) \tag{49}$$

where $\tilde{m}(x) \triangleq m(x)/\Lambda(x)$. □

Proof: Note that $\tilde{m}(x)$ has degree $\deg \tilde{m}(x) \geq k > \deg C^{(i)}(x)$. Note also that $\tilde{m}(x)$ divides $\gcd(E^{(i)}(x), m(x))$ and thus $\tilde{m}(x)$ divides $E^{(i)}(x)$. We then have

$$Y^{(i)}(x) \bmod \tilde{m}(x) = C^{(i)}(x) + E^{(i)}(x) \bmod \tilde{m}(x) \quad (50)$$
$$= C^{(i)}(x). \quad (51)$$

□

If the division in (48) (or in Proposition (3), respectively) does not work out, or if the resulting polynomials $C^{(i)}(x)$ do not satisfy $\deg C^{(i)}(x) < k$, then a decoding failure should be declared.

V. Proof of The Algorithm

A. Proof of Lemma 1

First, $\delta_{\max}(\Lambda') \geq \delta_{\max}(\Lambda'')$ is obvious from the assumptions. From (24), we obtain

$$r(x) \triangleq b^{(i)}(x)\Lambda(x) \bmod m^{(i)}(x) \quad (52)$$
$$= \kappa'' r'(x) - \kappa' x^{d'-d''} r''(x) \quad (53)$$

with

$$r'(x) \triangleq b^{(i)}(x)\Lambda'(x) \bmod m^{(i)}(x) \quad (54)$$
$$r''(x) \triangleq b^{(i)}(x)\Lambda''(x) \bmod m^{(i)}(x) \quad (55)$$

by the natural ring homomorphism $F[x] \to F[x]/m^{(i)}(x)$. It is then obvious from (53) that $\deg r(x) < \deg r'(x) = d'$, which is (25).

For the remaining proof, we define

$$\delta^{(\ell)}(\Lambda) \triangleq \mathrm{rd}^{(\ell)}(\Lambda) - \tau^{(\ell)} \quad (56)$$

for every $\ell \in \{1, \ldots, L\}$. With this notation, we have

$$\delta^{(i)}(\Lambda) < \delta^{(i)}(\Lambda') \quad (57)$$

from (25). We will next show that

$$\delta^{(j)}(\Lambda) \leq \delta^{(i)}(\Lambda') \text{ for } j < i \quad (58)$$

and

$$\delta^{(k)}(\Lambda) < \delta^{(i)}(\Lambda') \text{ for } k > i. \quad (59)$$

Clearly, (57)–(59) together imply both (26) and either (27) or (28) (or both).

To this end, we first note that $d' - d'' = \delta^{(i)}(\Lambda') - \delta^{(i)}(\Lambda'')$, and thus

$$d' - d'' + \delta^{(i)}(\Lambda'') = \delta^{(i)}(\Lambda'). \quad (60)$$

We then note from (24) that

$$\delta^{(\ell)}(\Lambda) \leq \max\left\{\delta^{(\ell)}(\Lambda'), d' - d'' + \delta^{(\ell)}(\Lambda'')\right\} \quad (61)$$

for every $\ell \in \{1, \ldots, L\}$.

Concerning (58), the assumption $i_{\max}(\Lambda') = i$ implies

$$\delta^{(j)}(\Lambda') \leq \delta^{(i)}(\Lambda') \quad (62)$$

for every $j < i$, and $i_{\max}(\Lambda'') = i$ implies

$$\delta^{(j)}(\Lambda'') \leq \delta^{(i)}(\Lambda''). \quad (63)$$

It then follows from (61)–(63) that for all $j < i$

$$\delta^{(j)}(\Lambda) \leq \max\left\{\delta^{(i)}(\Lambda'), d' - d'' + \delta^{(i)}(\Lambda'')\right\}, \quad (64)$$

and (58) follows from (60).

Concerning (59), the assumption $i_{\max}(\Lambda') = i$ implies

$$\delta^{(k)}(\Lambda') < \delta^{(i)}(\Lambda') \quad (65)$$

for every $k > i$, and $i_{\max}(\Lambda'') = i$ implies

$$\delta^{(k)}(\Lambda'') < \delta^{(i)}(\Lambda''). \quad (66)$$

It then follows from (61), (65), and (66) that for all $k > i$

$$\delta^{(k)}(\Lambda) < \max\left\{\delta^{(i)}(\Lambda'), d' - d'' + \delta^{(i)}(\Lambda'')\right\}, \quad (67)$$

and (59) follows from (60).

B. Assertions

For the detailed proof, we annotate the algorithm of Section III with some extra variables and some assertions as shown in Algorithm 2 on the next page. We now prove these assertions one by one, except that the proof of Assertion (A.1) is deferred to the end of this section.

Assertion (A.2) is obvious both from the initialization and from (A.11). Assertion (A.3) is the result of the **repeat** loop, as discussed at the beginning of Section III-A.

Assertion (A.4) is obvious. Assertions (A.5)–(A.8) follow from (A.2)–(A.4), followed by the swap in lines 21–23.

As for (A.9), when $b^{(i)}(x)$ is visited for the very first time (i.e., the first execution of line 26 for some index i), we have $d = \deg m^{(i)}(x)$ and $\mathrm{rd}^{(i)}(\Lambda) < d$ is obvious. For all later executions of line 26, we have $d = \mathrm{rd}^{(i)}(\Lambda)$ and $d^{(i)} = \mathrm{rd}^{(i)}(\Lambda^{(i)})$ before line 26, and $\mathrm{rd}^{(i)}(\Lambda) < d$ after line 26 follows from Lemma 1.

In order to prove (A.10) and (A.11), we need to understand how line 26 changes the degree of $\Lambda(x)$.

Lemma 2. Line 26 changes the degree of $\Lambda(x)$ only in iterations where lines 21–24 are executed. □

The proof is omitted.

If lines 21–24 are executed, then line 26 changes the degree of $\Lambda(x)$ to

$$\deg \Lambda^{(i)}(x) + d - d^{(i)} = \deg \Lambda_k(x) + \Delta_k, \quad (68)$$

which is (A.10). With (A.7), the left-hand side of (68) yields also (A.11).

It remains to prove (A.1). First, we note that (A.1) clearly holds when the **loop** is entered for the first time. But if (A.1) holds, then $\Lambda^{(i)}(x)$ in (A.6) satisfies

$$\deg \Lambda^{(i)}(x) = \sum_{j \neq i}^{L} \left(\deg m^{(j)}(x) - d^{(j)}\right) + \deg m^{(i)}(x) - d. \quad (69)$$

It then follows from (68) that $\Lambda(x)$ after line 26 satisfies

$$\deg \Lambda^{(i)}(x) + d - d^{(i)} = \sum_{j=1}^{L} \left(\deg m^{(j)}(x) - d^{(j)}\right), \quad (70)$$

Algorithm 2

Annotated SPI Algorithm

1 **for** $i = 1, \ldots, L$ **begin**
2 $\quad \Lambda^{(i)}(x) := 0$
3 $\quad d^{(i)} := \deg m^{(i)}(x)$
4 $\quad \kappa^{(i)} := \mathrm{lcf}\, m^{(i)}(x)$
5 **end**
6 $\Lambda(x) := 1$
7 $\delta := \max_{i \in \{1,\ldots,L\}} \left(\deg m^{(i)}(x) - \tau^{(i)} \right)$
8 $i := 1$

Extra:
$k := 0$ (E.1)

9 **loop begin**

Assertions:
$\deg \Lambda(x) = \sum_{i=1}^{L} \left(\deg m^{(i)}(x) - d^{(i)} \right)$ (A.1)
$\deg \Lambda(x) > \deg \Lambda^{(i)}(x), \quad i = 1, \ldots, L$ (A.2)

10 $\quad$ **repeat**
11 $\quad\quad$ **if** $i > 1$ **begin** $i := i - 1$ **end**
12 $\quad\quad$ **else begin**
13 $\quad\quad\quad$ **if** $\delta \leq 0$ **return** $\Lambda(x)$
14 $\quad\quad\quad$ $i := L$
15 $\quad\quad\quad$ $\delta := \delta - 1$
16 $\quad\quad$ **end**
17 $\quad\quad$ $d := \delta + \tau^{(i)}$
18 $\quad\quad$ $\kappa :=$ coefficient of x^d in
$\quad\quad\quad\quad b^{(i)}(x)\Lambda(x) \bmod m^{(i)}(x)$
19 $\quad$ **until** $\kappa \neq 0$

Assertion:
$i = i_{\max}(\Lambda)$, $\delta = \delta_{\max}(\Lambda) \geq 0$ (A.3)

20 $\quad$ **if** $d < d^{(i)}$ **begin**

Assertion:
$d^{(i)} > d = \delta + \tau^{(i)} \geq \tau^{(i)}$ (A.4)
Extras:
$k := k + 1$, $i_k \triangleq i$, $\Lambda_k(x) \triangleq \Lambda(x)$,
$\Delta_k \triangleq d^{(i)} - d$, $d_k \triangleq d^{(i)}$ (E.2)

21 $\quad\quad$ **swap** $(\Lambda(x), \Lambda^{(i)}(x))$
22 $\quad\quad$ **swap** $(d, d^{(i)})$
23 $\quad\quad$ **swap** $(\kappa, \kappa^{(i)})$
24 $\quad\quad$ $\delta := d - \tau^{(i)}$

Assertions:
$d > d^{(i)} \geq \tau^{(i)}$ (A.5)
$\deg \Lambda^{(i)}(x) > \deg \Lambda(x)$ (A.6)
$\deg \Lambda^{(i)}(x) > \deg \Lambda^{(j)}(x)$ for $j \neq i$ (A.7)
$i_{\max}(\Lambda^{(i)}) = i$, $\delta_{\max}(\Lambda^{(i)}) \geq 0$ (A.8)

25 $\quad$ **end**
26 $\quad \Lambda(x) := \kappa^{(i)}\Lambda(x) - \kappa x^{d-d^{(i)}} \Lambda^{(i)}(x)$

Assertions:
$\mathrm{rd}^{(i)}(\Lambda) < d = \delta + \tau^{(i)}$ (A.9)
$\deg \Lambda(x) = \Delta_k + \deg \Lambda_k(x)$ (A.10)
$\quad > \deg \Lambda^{(i)}(x), \quad i = 1, \ldots, L$ (A.11)

27 **end**

which is (A.1).

For later use, we also record the following fact from (E.2) and (A.10):

Proposition 4. The polynomials $\Lambda_k(x)$ defined in (E.2) satisfy $\deg \Lambda_1(x) = 0$ (since $\Lambda_1(x) = 1$) and

$$\deg \Lambda_k(x) > \ldots > \deg \Lambda_2(x) > \deg \Lambda_1(x) \tag{71}$$

with

$$\deg \Lambda_{t+1}(x) = \Delta_t + \deg \Lambda_t(x) \tag{72}$$

for $1 \leq t < k$. □

Finally, we note that the algorithm is guaranteed to terminate because every execution of the **repeat** loop (lines 10–19) strictly decreases $\delta_{\max}(\Lambda)$ or $i_{\max}(\Lambda)$ according to Lemma 1 and the swap in lines 21–23 strictly decreases $d^{(i)}$.

C. Proving the Minimality of the Returned $\Lambda(x)$

Let $\Lambda(x)$ be the polynomial that is returned by the algorithm. It is clear at this point that $\Lambda(x)$ satisfies (1) for all $i \in \{1, \ldots, L\}$. It remains to prove that no polynomial of smaller degree satisfies (1) for all i. The proof involves several steps.

Let $\Lambda_1(x), \Lambda_2(x), \ldots, \Lambda_K(x)$ be all polynomials $\Lambda_k(x)$ from (E.2) and note that $\deg \Lambda(x) > \deg \Lambda_K(x)$.

Lemma 3. Any nonzero $\tilde{\Lambda}(x) \in F[x]$ with $\deg \tilde{\Lambda}(x) < \deg \Lambda(x)$ can be uniquely written as

$$\tilde{\Lambda}(x) = \sum_{k=1}^{K} q_k(x)\Lambda_k(x) \tag{73}$$

with polynomials $q_k(x)$ such that

$$\deg q_k(x) < \deg \Lambda_{k+1}(x) - \deg \Lambda_k(x) \tag{74}$$

for $k = 1, \ldots, K-1$, and

$$\deg q_K(x) < \deg \Lambda(x) - \deg \Lambda_K(x). \tag{75}$$

□

The lemma is obvious from dividing $\tilde{\Lambda}(x)$ successively by $\{\Lambda_K(x), \Lambda_{K-1}(x), \ldots, \Lambda_1(x) = 1\}$.

In the following, we will work towards proving that any nonzero polynomial $\tilde{\Lambda}(x)$ as in Lemma 3 satisfies $\delta_{\max}(\tilde{\Lambda}) \geq 0$, which implies that $\tilde{\Lambda}(x)$ cannot not satisfy (1) for all $i \in \{1, \ldots, L\}$.

To this end, we need to study the values of $i_{\max}(q_k\Lambda_k)$ and $\delta_{\max}(q_k\Lambda_k)$. We begin by noting (from (A.3) and (E.2)) that

$$i_{\max}(\Lambda_k) = i_k \text{ and } \delta_{\max}(\Lambda_k) \geq 0 \tag{76}$$

for all $k \in \{1, \ldots, K\}$.

From Proposition 4, we have

$$\Delta_k = \deg \Lambda_{k+1}(x) - \deg \Lambda_k(x) \tag{77}$$

for $k \in \{1, \ldots, K-1\}$ and

$$\Delta_K = \deg \Lambda(x) - \deg \Lambda_K(x). \tag{78}$$

We then obtain from (74)–(78) that

$$\deg q_k(x) < \Delta_k \tag{79}$$

for all $k \in \{1, \ldots, K\}$.

On the other hand, we have from (E.2) that

$$\Delta_k = d_k - \deg\Big(b^{(i_k)}(x)\Lambda_k(x) \bmod m^{(i_k)}(x)\Big) \tag{80}$$

for $k = 1, 2, \ldots, K$. We then obtain from (79) and (80) that

$$\deg q_k(x) + \deg\Big(b^{(i_k)}(x)\Lambda_k(x) \bmod m^{(i_k)}(x)\Big) < d_k \tag{81}$$

for $k = 1, 2, \ldots, K$.

Lemma 4. For any nonzero $q_k(x)$, we have

$$i_{\max}(q_k\Lambda_k) = i_{\max}(\Lambda_k) \tag{82}$$

and

$$\delta_{\max}(q_k\Lambda_k) = \deg q_k(x) + \delta_{\max}(\Lambda_k). \tag{83}$$

□

Proof: For all $i \in \{1, \ldots, L\}$, we clearly have

$$\mathrm{rd}^{(i)}(q_k\Lambda_k) \le \deg q_k + \mathrm{rd}^{(i)}(\Lambda_k). \tag{84}$$

For i_k, however, we have

$$\mathrm{rd}^{(i_k)}(q_k\Lambda_k) = \deg q_k + \mathrm{rd}^{(i_k)}(\Lambda_k) \tag{85}$$

from (81) and since $d_k \le \deg m^{(i_k)}(x)$. The lemma then follows from $i_{\max}(\Lambda_k) = i_k$. □

In the next step, we partition the indices $k \in \{1, \ldots, K\}$ into sets $S_1, \ldots, S_L$ such that

$$k \in S_i \iff i_{\max}(\Lambda_k) = i_k = i. \tag{86}$$

We then write (73) as

$$\tilde{\Lambda}(x) = \sum_{i=1}^{L} \Big(\sum_{k \in S_i} q_k(x)\Lambda_k(x) \Big) \tag{87}$$

$$= \sum_{i=1}^{L} \tilde{\Lambda}^{(i)}(x) \tag{88}$$

with

$$\tilde{\Lambda}^{(i)}(x) \triangleq \sum_{k \in S_i} q_k(x)\Lambda_k(x). \tag{89}$$

Lemma 5. If $\tilde{\Lambda}^{(i)}(x)$ is nonzero, then

$$i_{\max}(\tilde{\Lambda}^{(i)}) = i \tag{90}$$

and

$$\delta_{\max}(\tilde{\Lambda}^{(i)}) \ge 0. \tag{91}$$

□

The proof is given below. Consider now the mapping

$$\varphi : \tilde{\Lambda}(x) \mapsto \big(\varphi_1(\tilde{\Lambda}), \ldots, \varphi_L(\tilde{\Lambda})\big) \tag{92}$$

where φ_i is the mapping

$$\tilde{\Lambda}(x) \mapsto r^{(i)}(x) \triangleq b^{(i)}(x)\tilde{\Lambda}(x) \bmod m^{(i)}(x) \tag{93}$$

$$\mapsto (r_0^{(i)}, \ldots, r_{\deg m^{(i)}(x)-1}^{(i)}) \tag{94}$$

$$\mapsto (r_{\tau^{(i)}}^{(i)}, \ldots, r_{\deg m^{(i)}(x)-1}^{(i)}). \tag{95}$$

Note that $\tilde{\Lambda}(x)$ satisfies (1) if and only if $\tilde{\Lambda}(x) \in \ker\varphi$. Since φ is linear, we have

$$\varphi(\tilde{\Lambda}) = \sum_{i=1}^{L} \varphi(\tilde{\Lambda}^{(i)}). \tag{96}$$

But (91) implies that $\varphi(\tilde{\Lambda}^{(i)})$ is nonzero if and only if $\tilde{\Lambda}^{(i)}(x)$ is nonzero, and (90) implies that the nonzero elements in the list $\varphi(\tilde{\Lambda}^{(1)}), \ldots, \varphi(\tilde{\Lambda}^{(L)})$ are linearly independent. It follows that (96) cannot be zero (for any nonzero $\tilde{\Lambda}(x)$ as in Lemma 3), which means that $\tilde{\Lambda}(x)$ does not satisfy (1) for all $i \in \{1, \ldots, L\}$.

Proof of Lemma 5:

We clearly have

$$\delta_{\max}(\tilde{\Lambda}^{(i)}) \le \max_{k \in S_i} \delta_{\max}(q_k\Lambda_k). \tag{97}$$

We show equality in (97) by showing that

$$\mathrm{rd}^{(i)}\Big(\sum_{k \in S_i} q_k(x)\Lambda_k(x) \Big) = \max_{k \in S_i} \mathrm{rd}^{(i)}(q_k\Lambda_k). \tag{98}$$

For any $k, k' \in S_i$ with $k < k'$, we have

$$\deg m^{(i)}(x) \ge d_k > d_{k'} \ge \tau^{(i)} \tag{99}$$

and

$$\mathrm{rd}^{(i)}(q_k\Lambda_k) \ge \mathrm{rd}^{(i)}(\Lambda_k) \ge d_{k'} \tag{100}$$

and

$$\mathrm{rd}^{(i)}(q_k\Lambda_k) = \deg q_k(x) + \mathrm{rd}^{(i)}(\Lambda_k) < d_k \tag{101}$$

from (85) and (81). It follows that

$$\max_{k \in S_i} \mathrm{rd}^{(i)}(q_k\Lambda_k) = \mathrm{rd}^{(i)}(q_\ell\Lambda_\ell) \tag{102}$$

with $\ell \triangleq \min\{k \in S_i : q_k(x) \ne 0\}$, and (98) is obvious. From (97) and (98), we also have (90), and (91) is obvious from (98)–(100). □

VI. Conclusion

We introduced the simultaneous partial-inverse problem, which is similar, but not identical, to the multi-sequence shift-register synthesis (MSSRS) problem. We proposed a new algorithm to solve this problem, which is similar, but not identical, to known algorithms for MSSRS. The proof of the proposed algorithm does not resemble existing proofs of MSSRS algorithms. We demonstrated the application of the algorithm to decoding interleaved Reed-Solomon codes as in [8], and we anticipate its application to decoding Reed-Solomon codes beyond half the minimum distance as in [7].

References

[1] J.-H. Yu and H.-A. Loeliger, "Reverse Berlekamp-Massey decoding," *IEEE Int. Symp. on Information Theory,* Istanbul, Turkey, July 7–12, 2013.

[2] G.-L. Feng and K. K. Tzeng, "A generalized Euclidean algorithm for multi-sequence shift-register synthesis," *IEEE Trans. Information Theory*, vol. 35, pp. 584–594, May 1989.

[3] G.-L. Feng and K. K. Tzeng, "A generalization of the Berlekamp-Massey algorithm for multisequence shift-register synthesis with applications to decoding cyclic codes," *IEEE Trans. Information Theory*, vol. 37, pp. 1274–1287, Sept. 1991.

[4] G. Schmidt and V. R. Sidorenko, "Multi-sequence linear shift-register synthesis: the varying length case," *IEEE Int. Symp. on Information Theory,* Seattle, USA, July 9–14, 2006.

[5] J. S. R. Nielsen, "Generalized multi-sequence shift-register synthesis using module minimisation," *IEEE Int. Symp. on Information Theory,* Istanbul, Turkey, July 7–12, 2013.

[6] G. Schmidt, V. R. Sidorenko, and M.Bossert, "Collaborative decoding of interleaved Reed-Solomon codes and concatenated codes designs," *IEEE Trans. Information Theory*, vol. 55, pp. 2991–3012, July 2009.

[7] G. Schmidt, V. R. Sidorenko, and M.Bossert, "Decoding Reed-Solomon codes beyond half the minimum distance using shift-register synthesis," *IEEE Int. Symp. on Information Theory,* Seattle, USA, July 9–14, 2006.

[8] R. M. Roth and P. O. Vontobel, "Coding for combined block-symbol error correction," *IEEE Trans. Information Theory*, vol. 60, pp. 2697–2713, May 2014.

[9] C. Haslach and A. J. H. Vinck, "A deoding algorithm with restrictions for array codes," *IEEE Trans. Information Theory*, vol. 45, pp. 2339–2344, Nov. 1999 (and correction in the same publication, vol. 47, p. 470, Jan. 2001).

[10] J. J. Metzner and E. J. Kapturowski, "A general decoding technique applicable to replicated file disagreement location and concatenated code decoding," *IEEE Trans. Information Theory*, vol. 36, pp. 911–917, July 1990.

[11] J. L. Massey, "Shift-register synthesis and BCH decoding," *IEEE Trans. Information Theory*, vol. 15, pp. 122–127, May 1969.

[12] R. E. Blahut, *Algebraic Codes for Data Transmission.* Cambridge University Press, Cambridge, UK, 2003.

[13] J.-H. Yu and H.-A. Loeliger, "On irreducible polynomial remainder codes," *IEEE Int. Symp. on Information Theory,* Saint Petersburg, Russia, July 31–Aug. 5, 2011.