Initialization of errata evaluator polynomial for simultaneous computation in Berlekamp-Massey for Reed-Solomon

This is a continuation of this post on SO.

I am trying to implement an errata (errors-and-erasures) decoder for Reed-Solomon. My current approach is to use Berlekamp-Massey (because it's the most common algorithm and there are a lot of optimizations tricks that are known, but the Euclidian algorithm is very similar and should be pretty much the same for my case: same initialization variables and same output, only the calculation method is different).

What I know:

- the common errors(-only) Berlekamp Massey to compute (only) the errors locator polynomial.
- how to extend standard BM to simultaneously compute the errors evaluator polynomial.
- how to extend standard BM to compute errata (errors-and-erasures) locator polynomial.

What I don't know: how to mix both extensions, to simultaneously compute the errata (errors-and-erasures) evaluator polynomial. In fact what I think I miss is the initialization value for the errata evaluator support polynomial A (see below).

To be clearer, here are some figures that describe the variations of BM that I cited above, from the book `"Algebraic codes for data transmission", Blahut, Richard E., 2003, Cambridge university press.` (readable here, chapters 7.4 and 7.5):

- Standard (errors-only and locator polynomial Lambda only) BM:

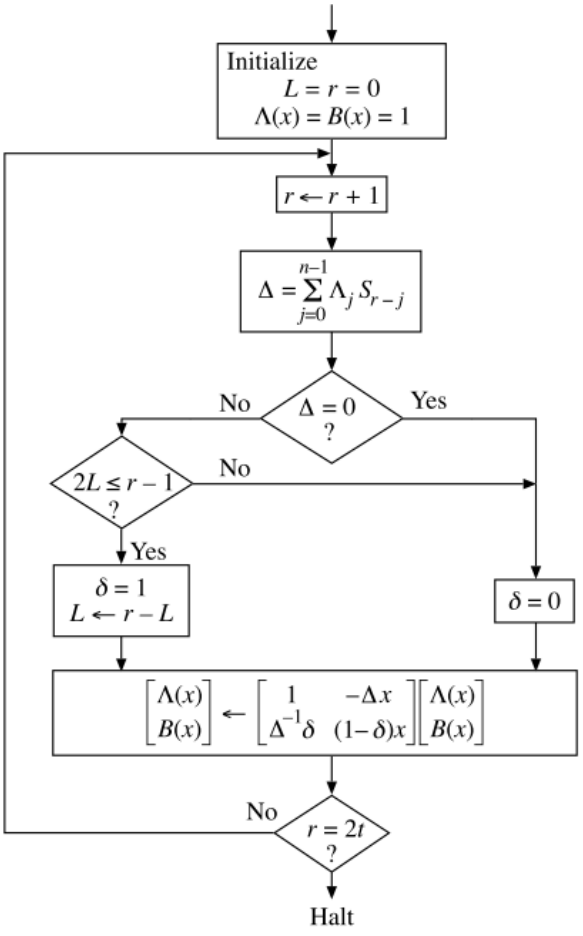

Figure 7.5. The Berlekamp–Massey algorithm

- BM extension to simultaneously compute the evaluator polynomial (named Gamma):

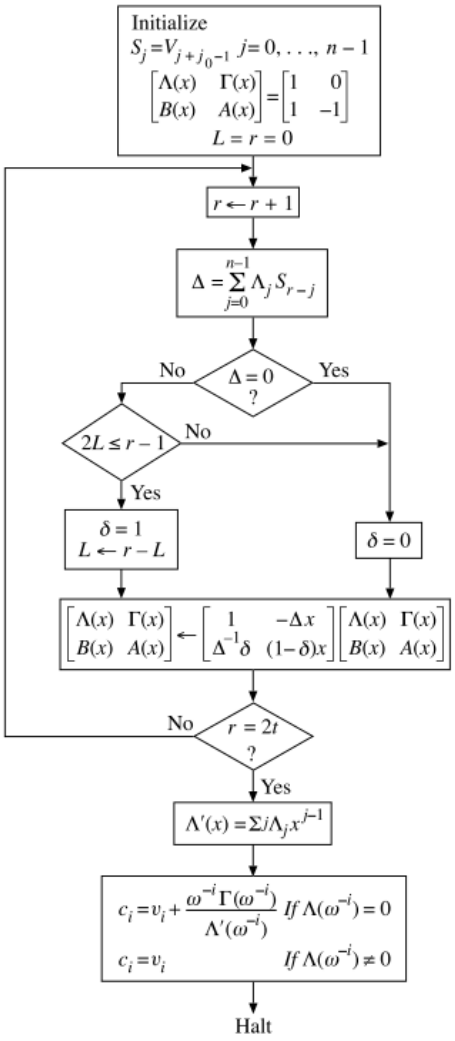

Figure 7.9. Another decoder using the Berlekamp algorithm and the Forney algorithm

- BM extension to decode errata (errors-and-erasures) but only locator polynomial:

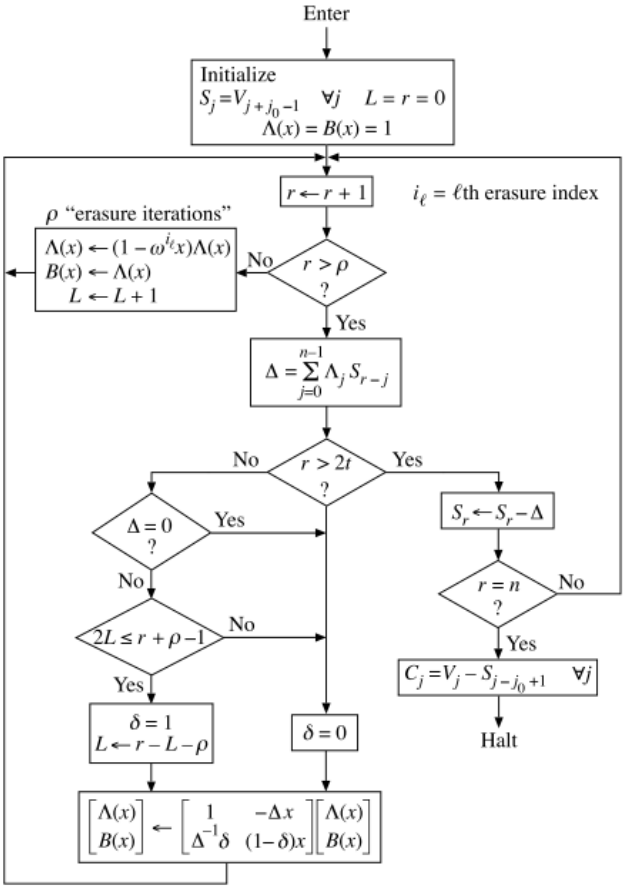

Figure 7.10. An error-and-erasure decoder for BCH codes

As of now, I just compute the erasures evaluator polynomial from the erasures locator polynomial, and I initialize Gamma = A = erasures evaluator polynomial (just like Lambda = B = erasures locator polynomial, see post on SO), but at the end of BM, the errata evaluator polynomial contains terms of too high order that should have been cancelled, for example:

```
Gamma:  34x^11 + 187x^10 + 200x^9 + 43x^5 + 129x^4 + 5x^3 + 62x^2 + 8x + 1
```

Instead of the correct:

```
Gamma:  43x^5 + 129x^4 + 5x^3 + 62x^2 + 8x + 1
```

I guess my initialization of the errata evaluator support polynomial A is wrong (when there's no erasures, Gamma = 1 and A = 0, here with erasures they are both initialized with the erasures evaluator polynomial), but I could not find any academic paper describing the initialization values.

The only paper I have found describing an errata decoder computing simultaneously the locator and evaluator polynomials is using the Euclidian algorithm instead of BM (from `"Efficient algorithms for decoding Reed-Solomon codes with erasures", Todd Mateer, Clemson University, Clemson, SC (2014)`):

Algorithm 4 : Efficient implementation of Key Equation solver

Input: The (possibly modified) syndrome polynomial $S^*(x) \in \mathbb{F}[x]$ for finite field $\mathbb{F}$;
Initialization polynomial $P(x)$ and optional second initialization polynomial $\Upsilon(x)$;
Starting step value K, stopping criteria Q, and integers $t, \mathrm{e} \geq 0$; Inverse flag (INV) equal to 0 or 1

Output: The polynomial $v(x)$ such that $r(x) = u(x) \cdot x^{2t+\mathrm{e}} + v(x) \cdot S^*(x)$
for some polynomials $u(x)$ and $r(x)$ where $\deg(r) < t$. [Optional: and polynomial $\Omega(x)$]

0. Allocate two arrays A and B each of size $2t + \mathrm{e}$ and initialized to all 0.
[Optional: Allocate two arrays Y and Z each of size $2t + \mathrm{e}$ and initialized to all 0.]
Set L be the degree of $P(x)$; Set $L_T = L$
Copy $A[i] = P_i$ (the degree i coefficient of P) and $B[i] = P_i$ for each i in $0 \leq i \leq L$.
[Opt: Copy $Y[i] = \Upsilon_i$ and $Z[i] = \Upsilon_i$ for each i in $0 \leq i < 2t + \mathrm{e}$.]
Set pointer V to the starting address of A and T to the starting address of B
[Opt: Set pointer Ω to the starting address of Y and Φ to the starting address of Z]
Assign $\psi := 1$ and $\gamma := 1$
If $(K - L \geq Q)$ then go to step 12
1. Assign $K := K + 1$.
2. Assign $D := \sum_{j=0}^{L} V[j] \cdot S^*[2t + \mathrm{e} + L - K - j]$.
NOTE: $S^*[i]$ is the degree i coefficient of $S^*(x)$ for all $i \geq 0$
3. If $D = 0$, then go to step 11.
4. Set $C := \psi \cdot D$.
If $2L < K$ then
5. Assign $T[j] := C \cdot T[j]$ for each j in $0 \leq j \leq L_T$
[Opt: Assign $\Phi[j] := C \cdot \Phi[j]$ for each j in $0 \leq j < 2t + \mathrm{e}$]
Then assign $T[j + K - 2L] := T[j + K + 2L] - \gamma \cdot V[j]$ for each j in $0 \leq j \leq L$.
[Opt: and assign $\Phi[j+K-2L] := \Phi[j+K+2L] - \gamma \cdot \Omega[j]$ for each j in $0 \leq j < 2t+\mathrm{e}+2L-K$.
6. Swap pointers T and V. [Opt: Swap pointers Φ and Ω]. Assign $L_T = L$
If INV=0, assign $\gamma := D$; If INV=1, assign $\psi := D^{-1}$.
7. Assign $L := K - L$.
8. else
9. If INV=0: Assign $V[j] := \gamma \cdot V[j]$ for each j in $0 \leq j \leq L$
If INV=0: [Opt: Assign $\Omega[j] := \gamma \cdot \Omega[j]$ for each j in $0 \leq j \leq 2t + \mathrm{e}$]
Assign $V[j + 2L - K] := V[j + 2L - K] - C \cdot T[j]$ for each j in $0 \leq j \leq L_T$
[Opt: and $\Omega[j + 2L - K] := \Omega[j + 2L - K] - C \cdot \Phi[j]$ for each j in $0 \leq j < 2t + \mathrm{e} - 2L + K$]
10. end if
11. If $(K - L < Q)$ then go to step 1
12. Return $v(x) = \{V[0], V[1], \ldots, V[L]\}$ [opt: and $\Omega(x) = \{\Omega[0], \Omega[1], \cdots, \Omega[2t + \mathrm{e}]\}$.]

But here it seems they are just using the same approach I did: initializing Gamma and A (here Omega and little omega) with the erasures evaluator polynomial, and they just do some trimming inside the conditional branches and at the return of the algo by limiting the indices of Omega.

Do someone know what values should Gamma and A be initialized with to avoid playing with the "too high order terms trimming hack"?

it.information-theory

edited 12 hours ago

asked yesterday

gaborous
116 4