Performance of ReedSolomon codes using the GuruswamiSudan algorithm with improved interpolation efficiency

Kwame Ackah Bohulu

January 4, 2019

1 Introduction

The idea of list decoding block codes was introduced by Elias [1] and Wozencraft [2] independently in the 1950s. In 1997, Sudan [3] applied this idea to decode low-rate (n, k) RS codes beyond the half-distance boundary $[(n-k-1)/2]$, where n is the code length and k is the message length. Later, Guruswami and Sudan [5, 6] improved the algorithm to decode RS codes of nearly any code rate beyond this boundary. Unfortunately, this algorithm remained impractical to implement until Kotter and Vardy [79] and Roth and Ruckenstein [10] presented lowcomplexity implementation methods for the key steps of the GS algorithm: interpolation and factorisation. In 2003, McEliece [11] gave an explicit tutorial discussion of the algorithm.

Traditional algebraic decoding algorithms for RS codes,including the Berlekamp-Massey algorithm [12] and Euclids algorithm [13, 14] generate a unique decoded codeword. They are very efficient in terms of running time but are unable to correct any number of errors greater than $[(n-k-1)/2]$ and therefore limits the performance of RS codes over deeply corruptive channels.

The GS algorithm for RS codes removes this limitation by finding a list of possible transmitted messages with decoding considered to be successful as long as the transmitted message is included in the list. The correct transmitted message is chosen by re-encoding the list of candidate messages and selecting the codeword with minimum distance to the received word. The GS algorithm improves the error correction capability significantly for low-rate (1/3) RS codes[5, 6]. However, for higher rate codes this algorithm can still improve the error correction capability but with a less significant improvement.

The GS algorithm has not been assessed by many researchers due to its high decoding complexity and it also requires a good understanding of mathematics. However, its greater error-correction capability makes it is a potential alternative decoding algorithm for RS codes and can be extended to the family of algebraicgeometric codes [5, 7], which lead to wider applications.

This paper describes the principle of the GS algorithm for RS codes from an algebraicgeometric point of view. Addressed towards improving the algorithms decoding efficiency, a novel complexity-reduced modification to the original algorithm is presented and a detailed complexity analysis is given

2 Overview of Guruswami - Sudan Algorithm

The table below shows the commonly used notations in this paper and their meanings

Symbol	Meaning
$\mathbb{F}_q$	finite field with q elements
$\mathbb{F}_q[x]$	ring of polynomials with coefficients from $\mathbb{F}_q$ and variable x
$\mathbb{F}_q[x^w]$	ring of polynomials from $\mathbb{F}_q$ with x degree $\leq w$
$\mathbb{F}_q[x, y]$	ring of nivariate polynomials with coefficients from $\mathbb{F}_q$ and variables x and y

Assuming a function $f(x)$ which is a subspace of $\mathbb{F}_q[x^{k-1}]$, a (n, k) RS code is generated by evaluating $f(x)$ at a set of points $x_0, x_1, ..., x_{n-1} \in \mathbb{F}_q$

$$(c_0, c_1, ..., c_{n-1}) = (f(x_0), f(x_1), ..., f(x_{n-1}))$$

it should be noted that

$$f(x) = f_0 + f_1 x + \cdots + f_{k-1} x^{k-1}$$

where the coefficients $f_0, f_1, ..., f_{k-1} \in \mathbb{F}_q$ are considered as the transmitted message.

2.1 Brief Description of the Guruswami-Sudan Algorithm

The above mentioned algorithm is made up of two major steps, namely Interpolation and Factorization.

- Interpolation: From the encoding process, it can be shown that evey member of the received word y_i was formed by a corresponding finite field element $x_i,\ 0 \leq i \leq n-1$. it is therefore possible to form $n(x_i, y_i)$ point pairs. In the interpolation step,we seek to construct a bivariate polynomial

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

  which has a minimal $(1, k-1)$- weight degree and has a zero order m over these n points. m which is called the multiplicity is the number of times the polynomial intersects the n points.

- Factorization: After the polynomial $Q(x, y)$ is found factorization is done to find the list L of possible transmitted message polynomials $p(x)$ and is given by

  $$L = \{p(x) : (y - p(x)) | Q(x, y) \text{ and } \deg(p(x)) < k\} \tag{2}$$

  The one with the minimum distance from the received message after re-encoding is chosen as the transmitted message.

2.2 Decoding Parameters

Necessary decoding parameters for understanding the GS algorithm are introduced here.

- We define the (u, v)-weight degree of monomial $x^i y^i$ as

  $$w - \deg_{u,v}(x^i y^j) = iu + jv \tag{3}$$

  This is used to sort monomials in a desired order. In this paper, monomials are arranged according to the $(1, k-1)$- reverse lexicographical ($(1, k-1)$-revlex) order. The sorting rule used is shown below

$$x^{i_1}y^{j_1} < x^{i_2}y^{j_2} \text{ if}$$
$$w-\deg_{1,k-1}(x^{i_1}y^{j_1}) < w-\deg_{1,k-1}(x^{i_2}y^{j_2})$$
$$\text{or } w-\deg_{1,k-1}(x^{i_1}y^{j_1}) = w-\deg_{1,k-1}(x^{i_2}y^{j_2}) \text{ and } i_1 > i_2$$

Using the decoding of the $(7,5)$ RS code as an example, the $(1,4)$ weight degree and $(1,4)$ revlex order monomials x^iy^j are shown in Tables 1 and 2. From table 2, we can see that $\mathrm{ord}(x^4) = 4, \mathrm{ord}(x^2y) = 9$ and $\mathrm{ord}(y^2) = 4$ and therefore $x^4 < x^2y < y^2$. $\mathrm{ord}(x^iy^i)$ is the $(1,k-1)$ revlex order of the monomial x^iy^i

- We define the weight degree of a nonzero bivariate polynomial $Q(x,y)$ as the weight degree of its leading monomial M_L ie

  $$Q(x,y) = a_0M_0+\cdots+a_LM_L, \text{ with } M_0 < \ldots < M_L \text{ and } a_0,\ldots,a_L \in \mathbb{F}_q, a_L \neq 0 \quad (4)$$

  and

  $$w-\deg_{1,k-1}(Q(x,y)) = w-\deg_{1,k-1}(M_L) \quad (5)$$

- $L = \mathrm{lod}(Q(X,Y)) = \mathrm{ord}(M_L)$ is called the leading order(lod) of $Q(X,Y)$. it is possible to compare two polynomials by comparing their lod

- $S_x(N)$ and $S_y(N)$ are denoted as the highest degree of x and y under the $(1,k-1)$- revlex order s.t.

  $$S_x(N) = \max\{i : \mathrm{ord}(x^iy^0) \leq N\} \quad (6)$$
  $$S_y(N) = \max\{i : \mathrm{ord}(x^0y^j) \leq N\} \quad (7)$$

  Where N is any non-negative integer

- it is possible to rewrite (5) as

  $$w-\deg_{1,k-1}(Q(x,y)) = S_x(L) \quad (8)$$

  since under the $(1,k-1)$-revlex order, x^iy^0 is the minimal monomial with weight degree i.

- The error correction capability t_m and the maximum number of candidate messages l_m with respect to a given multiplicity m are defined as

  $$t_m = n - 1 - \left\lfloor \frac{S_x(C)}{m} \right\rfloor \quad (9)$$

  $$l_m = S_y(C) \quad (10)$$

  $$C = n\binom{m+1}{2} \quad (11)$$

  The above parameters grow monotonically with multiplicity m

- $t_{m_{GS}} = n - 1 - \left\lfloor \sqrt{(k-1)n} \right\rfloor$ and is defined as the upper bound on the error-correcting capability of the GS algorithm. This is implies that there is also an upper bound on the value of m, (m_{GS}) $t_{m_{GS}}$ is greater than or equal to $\left\lfloor (n-k-1)/2 \right\rfloor$

Two examples are given to show how t_m and l_m grow with multiplicity m

3 Interpolation

In this section, the interpolation theorem is explained from the algebraic geometric point of view, followed by a detailed description of Kotter's interpolation algorithm and a modification to improve its efficiency.

3.1 Interpolation Theorem

In the case of RS codes $1, x, ..., x^a$ are the rational functions that have increasing pole order over the point of infinity p_∞ of a projective curve. In general, interpolated polynomials can be written as

$$Q(x, y) = \sum_{a,b \in \mathbb{N}} q_{ab} x^a y^b,\ q_{ab} \in \mathbb{F}_q \tag{12}$$

Again, functions $1, (1 - x_i), ..., (1 - x_i)^u$ are the rational functions that have increasing zero order over the finite-field element x_i used in encoding, and the received word $y_i \in \mathbb{F}_q$ It is possible to write the interpolated polynomial with respect to (x_i, y_i) as

$$Q(x, y) = \sum_{u,v \in \mathbb{N}} q_{uv}^{(x_i,y_i)} (x - x_i)^u (y - y_i)^v,\ q_{uv}^{(x_i,y_i)} \in \mathbb{F}_q \tag{13}$$

if $q_{uv}^{(x_i,y_i)} = 0$ for $u + v < m, Q(x, y)$ has a multiplicity of m over (x_i, y_i) Notice that

$$x^a = (x - x_i + x_i) = \sum_{a \geq u} \binom{a}{u} x_i^{(a-u)} (x - x_i)^u \tag{14}$$

and

$$y^a = (y - y_i + y_i) = \sum_{b \geq v} \binom{b}{v} y_i^{(b-v)} (y - y_i)^v \tag{15}$$

substitutiing (14) and (15) into (12) gives

$$Q(x, y) = \sum_{u,v} \sum_{a \geq u, b \geq v} q_{ab} \binom{a}{u} \binom{b}{v} x_i^{(a-u)} y_i^{(b-v)} (x - x_i)^u (y - y_i)^v \tag{16}$$

which means

$$q_{uv}^{(x_i,y_i)} = \sum_{a \geq u, b \geq v} q_{ab} \binom{a}{u} \binom{b}{v} x_i^{(a-u)} y_i^{(b-v)} \tag{17}$$

(17) is (u, v)-Hasse derivative evaluation on the point (x_i, y_i) of the polynomial $Q(x, y)$ defined by (112) [17, 20, 21]. Using $D(Q)$ to denote the Hasse derivative evaluation of $Q(x, y)$, (17) can be denoted as

$$D_{uv}Q(x_i, y_i) = \sum_{a \geq u, b \geq v} q_{ab} \binom{a}{u} \binom{b}{v} x_i^{(a-u)} y_i^{(b-v)} \tag{18}$$

From the above analysis, the interpolation of the GS algorithm can be generalised as: Find a minimal $(1, k-1)$ - weight degree polynomial $Q(x, y)$ that satisfies

$$Q(x, y) = \min\{Q(x, y) \in \mathbb{F}_q[x, y] | D_{u,v}Q(x_i, y_i) = 0 \text{ for } i = 0, ..., n-1 \text{ and } u+v < m, (u, v \in \mathbb{N})\} \tag{19}$$

3.2 Kotter's Algorithm

The algorithm suggested by Kotter[6-8] provides an efficient method forpolynomial reconstruction. It is an iterative modification algorithm based of these 2 properties of the Hasse Derivative

Property 1: Linear function of the Hasse derivative
If $H, Q \in \mathbb{F}_q[x, y]$, c_1 and $c_2 \in \mathbb{F}_q$, then

$$D(c_1H + c_2Q) = c_1D(H) + c_2D(Q) \tag{20}$$

Property 2: Bilinear Hasse Derivative If $H, Q \in \mathbb{F}_q[x, y]$, then

$$[H, Q]_D = HD(Q) - QD(H) \tag{21}$$

if the Hasse derivative evaluation of $D(Q) = d_1$ and $D(H) = d_2 (d_1, d_2 \neq 0)$, based on Property 1 it is obvious to conclude that

$$[H, Q]_D = 0 \tag{22}$$

If $\text{lod}(H) > \text{lod}(Q)$, the new constructed polynomial from (28) has leading order $\text{lod}(H)$. Therefore by performing the bilinear Hasse derivative over two polynomials both of which have nonzero evaluations, one can reconstruct a polynomial which has zero Hasse derivative evaluation. Based on this principle, Kotters algorithm is to iteratively modify a set of polynomials through all n points and with every possible (u, v) pair under each point.

With multiplicity m, there are $\binom{m+1}{2}$ pairs of (u, v). They are arranged as follows

$$(u, v) = (0, 0), (0, 1), ..., (0, m-1), (1, 0), ...(1, m-2), ..., (m-1, 0)$$

This means that there will be $C = n\binom{m+1}{2}$ iterative modifications required for a given RS code. The index for the iterative modifications is given by $i_k, i_k = 0, 1, ...C$

3.2.1 Procedure for Kotter Algorithm

- The initial step is initialize a group of polynomials G_0 (ie $i_k = 0$) as

$$G_0 = \{g_{0,j} = y^j, \quad j = 0, 1, ...l_m\} \tag{23}$$

  where l_m is defined in (10). It is important to point out that

$$g_{0,j} = \min\{g(x, y) \in \mathbb{F}_q[x, y] | \deg_y(M_L) = j\} \tag{24}$$

  Where M_L is the leading order of g

- Next each is tested using (18) by

$$\Delta_j = D_{i_k=0}(g_{i_k,j}) \tag{25}$$

  if $\Delta_j = 0$ no further modifications are required

- if $\Delta_j \neq 0$ the polynomial is modified using the Bilinear Hasse derivative defined in (21)

- To construct a group of polynomials which satisfy

$$\begin{aligned} g_{i_k+1,j} = \min \Bigg\{ & g(x, y) \in \mathbb{F}_q[x, y] | D_{i_k,j}(g_{i_k+1,j}) = 0, \\ & D_{i_k-1,j}(g_{i_k+1,j}) = 0, ..., D_{0,j}(g_{i_k+1,j}) = 0 \\ & \text{and } deg_y(M_L) = j \Bigg\} \end{aligned} \tag{26}$$

  we choose the minimal polynomial among those polynomials with $\Delta_j neq 0$, denote its index as j^* and record it as g^*

$$\begin{aligned} j^* &= \text{index}(\min g_{i_k,j} | \Delta_j \neq 0) \\ g^* &= g_{i_k,j^*} \end{aligned} \tag{27}$$

- For those polynomials with $\Delta_j \neq 0$ but $j \neq j^*$ modify them using the Bilinear Hasse derivative defined in (21) without increasing the leading order.

$$g_{i_k+1,j} = [g_{i_k,j}, g^*]_{D_{i_k}} \tag{28}$$

- for g^* we modify it (21) while increasing the leading order.

$$g_{i_k+1,j^*} = [xg^*, g^*]_{D_{i_k}} \tag{29}$$

- After C iterative modifications the minimal polynomial in G_C is the interpolated polynomial that satisfies (19), and it is chosen to be factorised in the next step