INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

An Alternative to Factorization: a Speedup for SUDAN*'s Decoding Algorithm and its Generalization to Algebraic-Geometric Codes*

Daniel Augot — Lancelot Pecquet

No 3532

Octobre 1998

THÈME 2

ISSN 0249-6399

INRIA
ROCQUENCOURT

An Alternative to Factorization: a Speedup for SUDAN's Decoding Algorithm and its Generalization to Algebraic-Geometric Codes

Daniel Augot — Lancelot Pecquet

Thème 2 — Génie logiciel
et calcul symbolique
Projet CODES

Rapport de recherche n°3532 — Octobre 1998 — 15 pages

Abstract: We propose an improvement to SUDAN's algorithm for decoding REED-SOLOMON codes beyond half of their minimum distance, and its generalisation to algebraic-geometric codes. Both algorithms, in their original version, involve factorisation of polynomials. The main idea consists in replacing factorisation by an iterative root finding procedure of low complexity based on NEWTON approximation. In the case of REED-SOLOMON codes we give real complexity of τ-reconstruction.

Key-words: SUDAN's algorithm, list-decoding, τ-reconstruction, complexity, (generalized) REED-SOLOMON codes, algebraic-geometric codes, factorisation of polynomials, NEWTON's approximation algorithm.

(Résumé : tsvp)

e-mail: Daniel.Augot@inria.fr, Lancelot.Pecquet@inria.fr

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
Téléphone : 01 39 63 55 11 - International : +33 1 39 63 55 11
Télécopie : (33) 01 39 63 53 30 - International : +33 1 39 63 53 30

Une alternative à la factorisation: accelération de l'algorithme de décodage de SUDAN et de sa généralisation aux codes géométriques

Résumé : Nous proposons une amélioration à l'algorithme de SUDAN pour décoder les codes de REED-SOLOMON au delà de la moitié de leur distance minimum, ainsi qu'à sa généralisation aux codes géométriques. La factorisation de polynômes intervient dans la version initiale de ces deux algorithmes. L'idée principale que nous présentons consiste à remplacer cette factorisation par une recherche itérative de racines, dont la complexité est faible, basée sur l'algorithme d'approximation de NEWTON. Grâce à cette modification, nous pouvons donner la complexité effective de la τ-reconstruction des codes de REED-SOLOMON.

Mots-clé : Algorithme de SUDAN, *list-decoding*, τ-reconstruction, complexité, codes de REED-SOLOMON (généralisés), codes géométriques, factorisation de polynômes, algorithme d'approximation de NEWTON.

1 Introduction

Let $\mathbf{F}_q$ be a finite field with q elements. For any code $C \subset \mathbf{F}_q^n$, we call the **τ-reconstruction ball**[1] of a vector $y \in \mathbf{F}_q^n$ with respect to C, the set of all words c of C that differ from y in at most τ positions (*i.e.* $\mathrm{d}(y,c) \leq \tau$ where d is the HAMMING distance). We will denote this set of codewords by $\mathrm{B}_\tau(y)$. If $t = \lfloor (d-1)/2 \rfloor$ is the correction capacity of C, for any $y \in \mathbf{F}_q^n$, a ball of radius t around y contains at most one codeword and hence in that case, t-reconstruction is decoding.

M. SUDAN has proposed in [Sud97a, Sud97b] a method to do this τ-reconstruction for $\tau \geq t$ for some REED-SOLOMON codes. M. A. SHOKROLLAHI and H. WASSERMAN have generalized SUDAN's algorithm to algebraic-geometric codes in [SW97, SW98]. Both algorithms, in their original version, involve factorization of polynomials. The first one needs to factorize bivariate polynomials over $\mathbf{F}_q$, the second requires factorization of univariate polynomials over an algebraic function field of one variable with field of constants $\mathbf{F}_q$.

The algorithm for REED-SOLOMON codes will be fully detailed though they are easily shown to be algebraic-geometric [Sti93, p. 51], while the one introduced for algebraic-geometric codes is theoretical since practical manipulation of functions is not considered here.

These kinds of algorithms are "interpolation algorithms", which seem fundamentally different from syndrome decoding, though some connections between them have been discussed by I. DUURSMA [Duu98].

This paper focuses on the algorithmics and the complexity of SUDAN's algorithm. We propose a speedup to these algorithms that avoids factorization, using a NEWTON approximation algorithm.

This new method allows to give a complexity of $\mathrm{O}(n^4/k)$ arithmetic operations on $\mathbf{F}_q$ to realize τ-reconstruction.

In section 2, we remind SUDAN's algorithm and the generalization, made by SHOKROLLAHI and WASSERMAN to algebraic-geometric codes. Then, in section 3 we describe our algorithm for REED-SOLOMON codes and we deduce from it an asymptotic upper bound on the complexity of the τ-reconstruction problem when SUDAN's algorithm applies. Eventually, we present in section 5 the version of our method for algebraic-geometric codes.

2 The basic algorithms

2.1 SUDAN's algorithm for REED-SOLOMON codes

First, let us recall what are REED-SOLOMON codes:

Definition 1 *Let $p = (p_1, \ldots, p_n)$ consisting of n distinct elements of $\mathbf{F}_q$, and L be the vector space of polynomials over $\mathbf{F}_q$ of degree lower than k. The* REED-SOLOMON code *of dimension k over p is the linear code [MS88]:*

$$C = \{(f(p_1), \ldots, f(p_n)),\ f \in L\}.$$

For these codes, let $y \in \mathbf{F}_q^n$, we replace the problem of finding the set $\mathrm{B}_\tau(y)$ of all codewords $c = (c_1, \ldots, c_n)$ such that $c_i \neq y_i$ for at most τ values of i, by the problem of finding the set $\mathrm{B}_\tau^\star(y)$ of all polynomials of L such that $f(p_i) = y_i$ for at least $n - \tau$ values of i. Hence we have:

$$f \in \mathrm{B}_\tau^\star(y) \quad \Longleftrightarrow \quad (f(p_1), \ldots, f(p_n)) \in \mathrm{B}_\tau(y).$$

Before to state SUDAN's theorem, we need to recall the definition of the weighted degree of a bivariate polynomial:

[1] The expression "τ-reconstruction" comes from [Sud97a]. The idea is close to the idea of "list-decoding", developed in [Eli57].

Definition 2 *Consider the bivariate polynomial:*

$$Q(X,Y) = \sum_{i,j} \alpha_{i,j} X^i Y^j.$$

*Let a, b be nonzero integers, the (a,b)-*weighted degree *of $Q(X,Y)$ is:*

$$\mathrm{wdeg}_{(a,b)} Q(X,Y) = \max\{ia + jb \mid \alpha_{i,j} \neq 0\}.$$

Theorem 1 (SUDAN, 1997) *Let $Q(X,Y) \in \mathbf{F}_q[X,Y]$, if:*

- $Q(X,Y) \neq 0$,
- $Q(p_i, y_i) = 0$, *for all* $i \in \{1, \ldots, n\}$,
- $\mathrm{wdeg}_{(1,k-1)} Q(X,Y) < n - \tau$,

then $(Y - f(X))$ divides $Q(X,Y)$ for all $f \in \mathrm{B}^\star_\tau(y)$.

Proof: Let f be a polynomial in $\mathrm{B}^\star_\tau(y)$, and consider $Q_f(X)$ to be the polynomial $Q(X, f(X))$. Since $\deg f < k$ and $\mathrm{wdeg}_{(1,k-1)} Q(X,Y) < n - \tau$, $\deg Q_f(X) < n - \tau$. Moreover $f(p_i) = y_i$ for at least $n - \tau$ values of i, so $Q_f(X)$ has at least $n - \tau$ roots. Hence $Q_f(X)$ is identically zero, *i.e.* $(Y - f(X))$ divides $Q(X,Y)$.

Provided $\tau \leq \tau_{\max}$ where $\tau_{\max}$ is the maximum number of errors that SUDAN's algorithm can correct [Sud97b] (*cf.* footnote of Proposition 6 on page 10), Q exists and can be found by solving a system of linear equations whose unknowns are the coefficients of Q.

So SUDAN's algorithm consists in three steps:

1 find $Q(X,Y)$ like in Theorem 1;

2 factorize $Q(X,Y)$ in $\mathbf{F}_q[X,Y]$;

3 get all factors of the form $(Y - f(X))$ and keep only the f's for which $f(p_i) = y_i$ for at least $n - \tau$ values of i.

In section 3, we propose an acceleration to steps **2** and **3** and show that their cost is, in fact less than the cost of finding $Q(X,Y)$.

Note 1 *One can use the same trick as in [Ret75] to apply* SUDAN*'s algorithm to generalized* REED-SOLOMON *codes in the following way: with the notations of Definition 1, if $v = (v_1, \ldots, v_n) \in \mathbf{F}_q^n$ consists in non-zero elements, we can define the* generalized REED-SOLOMON code *[MS88, p. 303]:*

$$C_v = \{(v_1 c_1, \ldots, v_n c_n),\ c \in C\}.$$

If $y = (y_1, \ldots, y_n) \in \mathbf{F}_q^n$, and we are looking for the set B of codewords $c \in C_v$ such that $\mathrm{d}(c,y) \leq \tau$, we can define $y' = (y_1/v_1, \ldots, y_n/v_n)$ and look for the list B' of codewords $c' \in C$ such that $\mathrm{d}(c', y') \leq \tau$. It is clear that

$$c' = (c'_1, \ldots, c'_n) \in B' \iff (c'_1/v_1, \ldots, c'_n/v_n) \in B.$$

2.2 SHOKROLLAHI and WASSERMAN generalization to AG-codes

First, let us recall what are algebraic-geometric codes:

Definition 3 *Let $K/\mathbf{F}_q$ be an algebraic function field in one variable of genus g, $P_1, \ldots, P_n$ be n pairwise distinct places of degree 1 of K, $P = P_1 + \cdots + P_n$, D a divisor whose support does not contain the P_i's, and L denote the vector space $L(D)$. Then the* algebraic-geometric code *with these parameters is the linear code [TS91, p. 266]:*

$$C = \{(f(P_1), \ldots, f(P_n)),\ f \in L\}.$$

For these codes, let $y \in \mathbf{F}_q^n$, we replace the problem of finding the set $\mathrm{B}_\tau(y)$ by the problem of finding the set $\mathrm{B}_\tau^\star(y)$ all functions of L such that $f(p_i) = y_i$ in at least $n - \tau$ values of i. Hence we have, as for REED-SOLOMON codes:

$$f \in \mathrm{B}_\tau^\star(y) \quad \Longleftrightarrow \quad (f(P_1), \ldots, f(P_n)) \in \mathrm{B}_\tau(y).$$

We now recall the main theorem of [SW97], which applies for $D = \alpha Q$, where α is a nonzero integer and Q a place of degree one. In this case, C has dimension $k = \alpha + g - 1$ and $d \geq n - \alpha$.

Theorem 2 (SHOKROLLAHI and WASSERMAN, 1997) *Let $\beta = \lceil \sqrt{2\alpha n} \rceil + g - 1$ and $b = \left\lfloor \frac{\lceil \sqrt{2\alpha n} \rceil}{\alpha} \right\rfloor$, then for $\tau = n - \beta - 1$, there exists a polynomial:*

$$G(Y) = \sum_{j=0}^{b} u_j Y^j \in K[Y],$$

such that:

- $G \neq 0$
- $G_{P_i}(y_i) = \sum_{j=0}^{b} u_j(p_i) y^i = 0$, *for all* $i \in \{1, \ldots, n\}$
- $u_j \in L((\beta - j\alpha)Q)$, *for all* $j \in \{1, \ldots, b\}$

In addition, $(Y - f)$ divides $G(Y)$ for all $f \in \mathrm{B}_\tau^\star(y)$.

Proof: *cf.* [SW97].

Like in SUDAN's algorithm, SHAKROLLAHI and WASSERMAN algorithm consists in three steps:

1 find $G(Y)$ like in Theorem 2;

2 factorize $Q(Y)$ in $K[Y]$;

3 get all factors of the form $(Y - f(X))$ and keep only the f's for which $f(P_i) = y_i$ for at least $n - \tau$ values of i.

In section 5, we propose an acceleration to steps **2** and **3**.

3 The speedup for Reed-Solomon codes

3.1 The algorithm

We do not need a complete factorization of the polynomials: we only need some roots. This can be done by a Newton approximation method.

We suppose that we have found $Q(X,Y)$ like in Theorem 1, of minimal degree in Y. For each index i such that $Q'(p_i, y_i) \neq 0$ — $Q'(X,Y)$ means $\frac{\partial Q(X,Y)}{\partial Y}$ — we can do a Newton approximation of a polynomial φ such that $\varphi(y_i) = 0$. In algorithm 1, described in figure 1, the list B is built and at the end of the algorithm, consists in all polynomials f of degree lower than k such that $f(p_i) = y_i$ for at least $n - \tau$ values of i, *i.e.* $f \in \mathrm{B}^\star_\tau(y)$.

$B \leftarrow \varnothing$
$N \leftarrow \lceil \log_2(k-1) \rceil$
$I \leftarrow \{i \in \{1, \ldots, n\} / \ Q'(p_i, y_i) \neq 0\}$
for $i \in I$ **do**
{
  $Q_i \leftarrow Q(X - p_i, Y)$ // *We want* $Q_i(0, y_i) = 0$
  $\varphi \leftarrow y_i$
  for j **from** 0 **to** $N-1$ **do**
  {
    $\varphi \leftarrow (\varphi - Q_i(X, \varphi(X))/Q_i'(X, \varphi(X))) \mod X^{2^{j+1}}$
  }
  $\varphi \leftarrow \varphi(X + p_i)$ // *We want* $\varphi(p_i) = y_i$
  if $\varphi(p_j) = y_j$ for at least $n - \tau$ values of j **then**
  {
    $B \leftarrow B \cup \{\varphi\}$
  }
}
return B

Figure 1: Algorithm 1, for Reed-Solomon codes

Proposition 1 *Let $Q(X,Y)$ like in Theorem 1, of minimal degree in Y, let I be the subset of $\{1, \ldots, n\}$ such that $Q'(p_i, y_i) \neq 0$, then*

1 *for any f such that $Q(X, f(X)) = 0$, there is at least one $i \in I$ such that $f(p_i) = y_i$;*

2 *if for some $i \in I$, $f_1(p_i) = f_2(p_i) = y_i$ then $f_1 = f_2$;*

3 *for any $i \in I$, algorithm 1 finds, if it exists the unique $f \in \mathrm{B}^\star_\tau(y)$ such that $f(p_i) = y_i$;*

hence algorithm 1 returns $\mathrm{B}^\star_\tau(y)$.

Proof:

1 As $Q(X, f(X)) = 0$, there exists $R(X,Y)$ such that

$$Q(X,Y) = (Y - f(X)) \cdot R(X,Y)$$

then

$$Q'(X,Y) = R(X,Y) + (Y - f(X)) \cdot R'(X,Y).$$

Let $S = \{i \in \{1,\dots,n\} / \ f(P_i) = y_i\}$ and $S' = \{i \in \{1,\dots,n\} / \ f(P_i) \neq y_i\}$. We have for any index $i \in S'$, $R(p_i, y_i) = 0$. If for any $i \in S$, $R(p_i, y_i) = 0$, then $R(X,Y)$ is a polynomial such that:

- $R(X,Y) \neq 0$,
- $R(p_i, y_i) = 0$, for all $i \in \{1,\dots,n\}$,
- $\mathrm{wdeg}_{(1,k-1)} R(X,Y) < n - \tau$,

which is not possible since $Q(X,Y)$ has minimal degree in Y among such polynomials. We therefore conclude that there exists $i \in S$ such that $R(p_i, y_i) \neq 0$. As $Q'(p_i, y_i) = R(p_i, y_i) \neq 0$, we have that $i \in I$.

2 If for some $i \in I$, $f_1(p_i) = f_2(p_i) = y_i$, then there exists $R(X,Y)$ such that

$$Q(X,Y) = (Y - f_1(X))(Y - f_2(X)) \cdot R(X,Y)$$

and then

$$\begin{aligned} Q'(X,Y) =& (Y - f_2(X)) \cdot R(X,Y) \\ &+ (Y - f_1(X)) \cdot R(X,Y) \\ &+ (Y - f_1(X))(Y - f_2(X)) \cdot R'(X,Y) \end{aligned}$$

which implies $Q'(p_i, y_i) = 0$: it is impossible since $i \in I$.

3 Suppose there exists $f \in \mathrm{B}^{\star}_{\tau}(y)$ such that $f(p_i) = y_i$. From item 1, we can suppose $i \in I$. We will write φ_j the value of φ at the beginning of the j-th **for** loop. We show by recurrence that for any $j \geq 0$, $f(X) = \varphi_j(X) \mod X^{2^j}$.

The initialization is for $j = 0$ we do have $\varphi_0 = y_i$, *i.e.* $\varphi_0 = y_i \mod X^{2^0}$. Suppose now that the result is true until the j-th rank, we will write $\widetilde{Q}(Y) \in \mathbf{F}_q[[X]][Y]$ the univariate polynomial in Y corresponding to $Q(X,Y)$ with coefficients in the ring $\mathbf{F}_q[[X]]$ of univariate power series over $\mathbf{F}_q$. We can take the 2nd order TAYLOR development of $\widetilde{Q}$ around φ_j, *i.e.* there exists $R \in \mathbf{F}_q[[X]][Y]$ such that

$$\begin{aligned} \widetilde{Q}(Y) = \widetilde{Q}(\varphi_j) &+ (Y - \varphi_j) \cdot \widetilde{Q}'(\varphi_j) \\ &+ (Y - \varphi_j)^2 \cdot R(Y - \varphi_j). \end{aligned}$$

Since the recurrence hypothesis, $\varphi_j(p_i) = f(p_i) = y_i$ and $Q(p_i, y_i) \neq 0 \Longrightarrow \widetilde{Q}'(\varphi_j) \neq 0$, we can therefore divide the previous equality by $\widetilde{Q}'(\varphi_j)$ to get after specialization in $Y = f$:

$$\begin{aligned} 0 = \frac{\widetilde{Q}(f)}{\widetilde{Q}'(\varphi_j)} = \frac{\widetilde{Q}(\varphi_j)}{\widetilde{Q}'(\varphi_j)} &+ (f - \varphi_j) \\ &+ (f - \varphi_j)^2 \cdot \frac{R(f - \varphi_j)}{\widetilde{Q}'(\varphi_j)}, \end{aligned}$$

which implies

$$f = \underbrace{\varphi_j - \frac{\widetilde{Q}(\varphi_j)}{\widetilde{Q}'(\varphi_j)}}_{\varphi_{j+1}} - (f - \varphi_j)^2 \cdot \frac{R(f - \varphi_j)}{\widetilde{Q}'(\varphi_j)}.$$

From the recurrence hypothesis, we know that $f - \varphi_j = 0 \mod X^{2^j}$, hence $(f-\varphi_j)^2 = 0 \mod X^{2^{j+1}}$. We now have proved that for any $j \geq 0$, $f(X) = \varphi_j(X) \mod X^{2^j}$.

After N loops, $f = \varphi_N \mod X^k$. As $\deg f < k$, $f = \varphi_N$.

To check if $f \in \mathrm{B}^\star_\tau(y)$, one just have to check that $S = \{i \in \{1, \ldots, n\} / \ f(P_i) = y_i\}$ has at least $n - \tau$ elements; if it is the case, as we know $\deg f < k$, *i.e.* $f \in L$, and $Q(X, f(X)) = 0$ from Theorem 1.

3.2 A tip for implementation: avoid doubles

Once we have found f such that $(Y - f(X))$ divides $Q(X,Y)$, we can remove from our investigation all indexes $i \in I$ such that $f(p_i) = y_i$. This comes from the following proposition, which results directly from Proposition 1:

Proposition 2 *If*

$$\{i \in I / \ f_1(p_i) = y_i\} \cap \{i \in I / \ f_2(p_i) = y_i\} \neq \varnothing$$

then

$$\{i \in I / \ f_1(p_i) = y_i\} = \{i \in I / \ f_2(p_i) = y_i\}$$

and $f_1 = f_2$.

So if we have found φ such that $\varphi(p_j) = y_j$ for at least $n - \tau$ values of j, we can add the line:

"remove from I all $j \in \{1, \ldots, n\}$ such that $\varphi(p_j) = y_j$"

because the algorithm would return the same φ on such p_j's.

3.3 Complexity

Proposition 3 *If* $\deg_Y Q(X,Y) = b$, *then the inner* ***for*** *loop of algorithm 1 takes* $\mathrm{O}(b \cdot \mathrm{m}(k))$ *arithmetic operations in* $\mathbf{F}_q$, *where* $\mathrm{m}(k)$ *is the cost of multiplication of two polynomials of degree* k *over* $\mathbf{F}_q$ *in the worst case.*

Proof: Let us consider $\widetilde{Q}(Y) \in K[Y]$, the univariate polynomial in Y with coefficients in $K = \mathbf{F}_q[[X]]$ corresponding to $Q(X,Y)$. and let us denote by φ_j, the value of φ considered as an element of K at the beginning of the j-th **for** loop. Then, there exists $R(Y) \in K[Y]$ such that:

$$\widetilde{Q}(Y) = \widetilde{Q}(\varphi_j) + (Y - \varphi_j) \cdot R(Y). \tag{1}$$

Similarly, there exists $S(Y) \in K[Y]$ such that

$$R(Y) = R(\varphi_j) + (Y - \varphi_j) \cdot S(Y).$$

But by derivation of equation 1, and specialization in φ_j, we have that $\widetilde{Q}'(\varphi_j) = R(\varphi_j)$. Hence

$$R(Y) = \widetilde{Q}'(\varphi_j) + (Y - \varphi_j) \cdot S(Y); \tag{2}$$

equation 1 shows that the quotient of the Euclidean division of $\widetilde{Q}(Y)$ by $(Y - \varphi_j)$ is $R(Y)$ and its remainder is $\widetilde{Q}(\varphi_j)$. As we divide by a polynomial of degree 1, this step is done in b multiplications

in K. Furthermore, equation 2 shows that $\widetilde{Q}'(\varphi_j)$ can be computed as the remainder of the Euclidean division of $R(Y)$ by $(Y-\varphi_j)$. As $\deg R = b-1$, and we again divide by a polynomial of degree 1, this step is done in $b-1$ multiplications in K. Altogether, we do $2b-1$ multiplications in K but as $\deg \varphi_j = 2^j = \frac{k-1}{2^{N-j}}$, each multiplication can be done in $\mathrm{m}\left(\frac{k-1}{2^{N-j}}\right)$. As $\mathrm{m}(l) \geq 2\mathrm{m}(l/2)$, hence to do to complete all N loops we need:

$$(2b-1)\sum_{j=0}^{N-1} \mathrm{m}\left(\frac{k-1}{2^{N-j}}\right) \leq \underbrace{(2b-1)\cdot \mathrm{m}(k-1)\sum_{j=0}^{N-1}\frac{1}{2^j}}_{\mathrm{O}(b\cdot \mathrm{m}(k))}.$$

Proposition 4 *Let $Q(X,Y) \in \mathbf{F}_q[X,Y]$ be as in Theorem 1, then*

$$\deg_Y Q(X,Y) \leq \left\lfloor \frac{n-\tau-1}{k-1} \right\rfloor.$$

As a consequence, the set $\mathrm{B}_\tau(y)$ contains at most $\left\lfloor \frac{n-\tau-1}{k-1} \right\rfloor$ codewords.

Proof: The condition $\mathrm{wdeg}_{(1,k-1)} Q(X,Y) < n-\tau$ implies that $(k-1)\deg_Y Q(X,Y) \leq n-\tau-1$ which is the first statement. Consequently, the number of polynomials $f(X)$ such that $Q(X,f(X))=0$ cannot exceed $\deg_Y Q(X,Y)$. As all polynomials of $\mathrm{B}^\star_\tau(y)$ realize this condition, and since $\mathrm{B}_\tau(y) = \mathrm{B}^\star_\tau(y)$, we have our result.

Proposition 5 *Algorithm 1 requires at most $\mathrm{O}\left(kn^2\right)$ arithmetic operations in $\mathbf{F}_q$.*

Proof: In the worst case, we have for each $i \in I$ a different solution, and $|I| = n$, hence we will loop n times. In the "NEWTON step" of algorithm 1, $\deg_Y Q$ is at most $\left\lfloor \frac{n-\tau-1}{k-1} \right\rfloor$ as we proved in Proposition 4. Moreover, the multiplication of two polynomials of degree lower than k can be done in $\mathrm{O}(k^2)$ arithmetic operations in $\mathbf{F}_q$. We therefore know that the total running time for this step will be $\mathrm{O}(n(n-\tau)k)$ from Proposition 3. The test "$f(p_j) = y_j$ for at least $(n-\tau)$ values of j" costs in the worst case n evaluations of f, whose degree is lower than k. Each evaluation can be done in $\mathrm{O}(k)$ by HÖRNER's rule [Knu98, p. 496]. So this step requires at most $\mathrm{O}(n^2k)$ operations.

Note 2 *There is a faster way to multiply two polynomials of degree k using the Fast* FOURIER *Transform (F.F.T.) which is in $\widetilde{\mathrm{O}}(k)$ operations*[2] *on $\mathbf{F}_q$. However this complexity is rather theoretical since one have to find a primitive root of unity one some extension of $\mathbf{F}_q$, which can be practically more expensive than basic multiplication. The reader can refer for instance to [BCS96]. Anyway, in this case "Newtonian loops" cost for each $i \in I$ $\widetilde{\mathrm{O}}((n-\tau))$ and the tests "$f(p_j) = y_j$ for at least $n-\tau$ values of j" cost each $\widetilde{\mathrm{O}}(n)$ with the method of evaluation in multiple points exposed in [CLR94, p. 786]. As in the worst case we need to loop n times, the global complexity becomes $\widetilde{\mathrm{O}}(n^2)$ arithmetic operations in $\mathbf{F}_q$.*

[2] The *soft O* notation is to be understood in the following way: "$A = \widetilde{\mathrm{O}}(m)$" means: "for some c, $A = \mathrm{O}(m\log^c m)$"

Let us compare the complexity of our method to bivariate factorization. In [Zip93, p. 339], an algorithm to factorize in $\mathbf{F}_q[X,Y]$ a squarefree primitive polynomial whose degree in X and degree in Y are bounded by m in $\mathrm{O}(m^7)$ arithmetic operations on some algebraic extension of $\mathbf{F}_q$. The condition $\mathrm{wdeg}_{(1,k-1)}Q(X,Y) < n-\tau$, allows $\deg_X Q(X,Y) = n-\tau$ and Proposition 4 allows at most $\deg_Y Q(X,Y) = \left\lfloor \frac{n-\tau-1}{k-1} \right\rfloor$. We therefore have $m = n-\tau$ and a complexity of $\mathrm{O}((n-\tau)^7)$ operations on some algebraic extension of $\mathbf{F}_q$, provided $Q(X,Y)$ is squarefree and primitive. Furthermore, factorization algorithms are much more complicated than the present one.

4 Complexity of Sudan's algorithm

Proposition 6 *Let C be a $[n,k]$ generalized* Reed-Solomon *code over $\mathbf{F}_q$. Provided $\tau \leq \tau_{\max}$, where $\tau_{\max}$ is the maximum number of errors that* Sudan*'s algorithm can correct,*[3] *for any $y \in \mathbf{F}_q^n$, one can find the list of all codewords closer than τ from y in $\mathrm{O}(n^4/k)$ arithmetic operations in $\mathbf{F}_q$.*

Proof: From [Sud97b], we know that when $\tau \leq \tau_{\max}$, there exists a polynomial $Q(X,Y)$ that realizes the conditions of Theorem 1. It can be found by solving over $\mathbf{F}_q$ a linear system of n equations with N unknowns — the coefficients of Q — with

$$N = \left\lfloor (r+1)\left((n-\tau) - \frac{k-1}{2}r\right)\right\rfloor$$

and

$$r = \left\lfloor \frac{n-\tau-2}{k-1} \right\rfloor .$$

This system can be solved by Gauss elimination [Sed89, p. 535] in $\mathrm{O}(n^2N)$ operations. As $N = \mathrm{O}(n^2/k)$, this first step costs $\mathrm{O}(n^4/k)$. We have shown in Proposition 5 that steps 2 and 3 of Sudan's algorithm can be done in $\mathrm{O}(n^2k)$ operations.

5 The speedup for algebraic-geometric codes

5.1 The algorithm

The Newton's approximation algorithm works more generally on a valuation ring [Lan95, pp. 493–494] and the previous method can be easily extended to algebraic-geometric codes. Actually, in this case, the formal description is even simpler than for Reed-Solomon codes.

We suppose that we have found $G(Y)$ like in Theorem 2, of minimal degree. For each index i such that $G'_{P_i}(y_i) \neq 0$, we can apply Newton's method to get a function φ such that $\varphi(y_i) = 0$. In algorithm 2, described in figure 2, the list B is built and at the end of the algorithm, consists in all functions f in $\mathrm{L}(\alpha Q)$ such that $f(P_i) = y_i$ for at least $n-\tau$ values of i, *i.e.* $f \in \mathrm{B}^\star_\tau(y)$. It is worthy of note that algorithm 2 returns only functions which are in $\mathrm{L}(\alpha Q)$ while factorization can give *a priori* some factors that belong to K but do not belong to $\mathrm{L}(\alpha Q)$.

[3] Sudan proves in [Sud97b] that $\tau_{\max} \geq \left(1 - \sqrt{2k/n} - \mathrm{o}(1)\right) n$.

$B \leftarrow \varnothing$
$N \leftarrow \lceil \log_2(\alpha+1) \rceil$
$I \leftarrow \{i \in \{1,\ldots,n\} / \ G'_{P_i}(y_i) \neq 0\}$
for $i \in I$ **do**
{
  $\varphi \leftarrow y_i$
  for j **from** 0 **to** $N-1$ **do**
  {
    $\varphi \leftarrow \varphi - G(\varphi)/G'(\varphi)$
  }
  if $\varphi(P_j) = y_j$ for at least $n-\tau$ values of j **then**
  {
    $B \leftarrow B \cup \{\varphi\}$
  }
}
return B

Figure 2: Algorithm 2, for algebraic-geometric codes

Proposition 7 *Let $G(Y)$ like in Theorem 2, of minimal degree in Y, let I be the subset of $\{1,\ldots,n\}$ such that $G'_{P_i}(y_i) \neq 0$, then*

1 *for any root f of $G(Y)$, there is at least one $i \in I$ such that $G'_{P_i}(y_i) \neq 0$ and $f(P_i) = y_i$;*

2 *if for some $i \in I$, $f_1(P_i) = f_2(P_i) = y_i$ then $f_1 = f_2$;*

3 *for any $i \in I$, algorithm 2 finds, if it exists the unique $f \in \mathrm{B}^\star_\tau(y)$ such that $f(p_i) = y_i$;*

hence algorithm 2 returns $\mathrm{B}^\star_\tau(y)$.

Proof:

1 If $G(f) = 0$ then there exists a polynomial $R(Y)$ such that

$$G(Y) = (Y - f) \cdot R(Y).$$

Then

$$G'(Y) = R(Y) + (Y - f) \cdot R'(Y).$$

Let $S = \{i \in \{1,\ldots,n\} / \ f(P_i) = y_i\}$ and $S' = \{i \in \{1,\ldots,n\} / \ f(P_i) \neq y_i\}$. We have for any index $i \in S'$, $R_{P_i}(y_i) = 0$. As we have

$$\sum_{j=0}^{b} u_j Y^j = (Y-f)\sum_{j=0}^{b-1} r_j Y^j = \sum_{j=0}^{b} (r_{j-1} - r_j f) Y^j,$$

with $r_{-1} = 0$. So we have for all $j \in \{0,\ldots,b\}$,

$$r_j = \frac{u_j - r_{j-1}}{f} \tag{3}$$

with $u_j \in \mathrm{L}((\beta-\alpha j)Q)$ and $f \in \mathrm{L}(\alpha Q)\setminus\{0\}$. Let us show by recurrence that for all $j \geq 0$,

$$r_j \in \mathrm{L}((\beta-\alpha(j+1))Q). \tag{4}$$

First, $r_0 = u_0/f$; in this case,

$$\mathrm{val}_Q(r_0) = \mathrm{val}_Q(u_0) - \mathrm{val}_Q(f) \geq -\beta+\alpha$$

so $r_0 \in \mathrm{L}((\beta-\alpha)Q)$. Suppose know that statement 4 is true for any $j' \leq j$. We can deduce from equation 3 and recurrence hypothesis that

$$\begin{aligned}\mathrm{val}_Q(r_j) &\geq \min\left(\mathrm{val}_Q(u_j), \mathrm{val}_Q(r_{j-1})\right) - \mathrm{val}_Q(f)\\ &\geq \alpha j - \beta - \alpha = \alpha(j+1) - \beta.\end{aligned}$$

It is what we wanted to proof. Now, as

$$r_j \in \mathrm{L}((\beta-\alpha(j+1))Q) \subset \mathrm{L}((\beta-\alpha j)Q),$$

if for any $i \in S$, $R_{P_i}(y_i) = 0$, then $R \in K[Y]$ is a polynomial such that:

$$\begin{cases} r_j \in \mathrm{L}((\beta-j\alpha)Q), & 1 \leq j \leq b-1;\\ G_{P_i}(y_i) = \sum_{j=0}^{b} r_j(P_i)y^i = 0, & 1 \leq i \leq n;\end{cases}$$

which is not possible since G has minimal degree among such polynomials. We therefore conclude that there exists $i \in S$ such that $R_{P_i}(y_i) \neq 0$. As $G'_{P_i}(y_i) = R_{P_i}(y_i) \neq 0$, we have that $i \in I$.

2 If for some $i \in I$, $f_1(P_i) = f_2(P_i) = y_i$, then there exists $R(Y)$ such that

$$G(Y) = (Y-f_1)\cdot(Y-f_2)\cdot R(Y)$$

and then

$$\begin{aligned}G'(Y) = (Y-f_2)\cdot R(Y) &+ (Y-f_1)\cdot R(Y)\\ &+ (Y-f_1)\cdot(Y-f_2)\cdot R'(Y)\end{aligned}$$

which implies $G'_{P_i}(y_i) = 0$: it is impossible since $i \in I$.

3 Suppose there exists $f \in \mathrm{B}^{\star}_{\tau}(y)$ such that $f(p_i) = y_i$. From item 1, we can suppose $i \in I$. We will write φ_j the value of φ at the beginning of the j-th **for** loop. We show by recurrence that for any $j \geq 0$, $\mathrm{val}_{P_i}(\varphi_j - f) \geq 2^j$.

The initialization is for $j = 0$ we do have $\varphi_0 = y_i$ *i.e.* $\mathrm{val}_{P_i}(\varphi_0 - y_i) \geq 1 = 2^0$. Suppose now that the result is true until the j-th rank, we can take the 2nd order Taylor development of G around φ_j, *i.e.* there exists $R \in K[Y]$ such that

$$\begin{aligned}G(Y) = G(\varphi_j) &+ (Y-\varphi_j)\cdot G'(\varphi_j)\\ &+ (Y-\varphi_j)^2\cdot R(Y-\varphi_j).\end{aligned}$$

Since the recurrence hypothesis, $\varphi_j(P_i) = f(P_i) = y_i \Longrightarrow G'_{P_i}(\varphi_j(P_i)) = G'_{P_i}(y_i) \neq 0$ and so $G'(\varphi_j) \neq 0$, we can therefore divide the previous equality by $G'(\varphi_j)$ to get after specialization in $Y = f$:

$$\begin{aligned}0 = \frac{G(f)}{G'(\varphi_j)} = \frac{G(\varphi_j)}{G'(\varphi_j)} &+ (f-\varphi_j)\\ &+ (f-\varphi_j)^2\cdot\frac{R(f-\varphi_j)}{G'(\varphi_j)},\end{aligned}$$

which implies

$$\underbrace{\varphi_j - \frac{G(\varphi_j)}{G'(\varphi_j)}}_{\varphi_{j+1}} - f = (f-\varphi_j)^2 \cdot \frac{R(f-\varphi_j)}{G'(\varphi_j)},$$

so

$$\begin{aligned}\mathrm{val}_{P_i}(\varphi_{j+1}-f) = 2\mathrm{val}_{P_i}(f-\varphi_j) + \mathrm{val}_{P_i}R(f-\varphi_j) \\ - \mathrm{val}_{P_i}G'(\varphi_i).\end{aligned}$$

As $R(Y)$'s coefficients are polynomials in terms of $u_m \in \mathrm{L}((\beta - m\alpha)Q)$, they have no pole in P_i so $\mathrm{val}_{P_i}R(f-\varphi_j) \geq 0$; in addition $\mathrm{val}_{P_i}G'(\varphi_j) = 0$ as $G'_{P_i}(\varphi_j(P_i)) \neq 0$. The recurrence hypothesis states $\mathrm{val}_{P_i}(f-\varphi_j) \geq 2^j$ and we can conclude that $\mathrm{val}_{P_i}(f-\varphi_{j+1}) \geq 2^{j+1}$.
We now have proved that for any $j \geq 0$, $\mathrm{val}_{P_i}(f-\varphi_j) \geq 2^j$. After N loops, $\mathrm{val}_{P_i}(f-\varphi_N) \geq \alpha+1$.

Let us show by recurrence that $\varphi_j \in \mathrm{L}(\alpha Q)$, for all $j \geq 0$. It is true for $\varphi_0 = y_i$. Suppose that $\varphi_{j'} \in \mathrm{L}(\alpha Q)$ for all $j' \leq j$. We have that

$$\varphi_{j+1} = \varphi_j - \frac{G(\varphi_j)}{G'(\varphi_j)};$$

since for any $l \in \{0, \ldots, b\}$, $u_l \in \mathrm{L}((\beta - \alpha l)Q)$,

$$\begin{cases} G(\varphi_j) = \sum_{l=0}^{b} u_l \varphi_j^l & \in \mathrm{L}(\beta Q); \\ G'(\varphi_j) = \sum_{l=0}^{b} l u_l \varphi_j^{l-1} & \in \mathrm{L}((\alpha-\beta)Q); \end{cases}$$

we deduce that the only pole of $\frac{G(\varphi_j)}{G'(\varphi_j)}$ is Q. Furthermore

$$\begin{cases} G(\varphi_j) \in \mathrm{L}(\beta Q) & \Longrightarrow \quad \mathrm{val}_Q(G(\varphi_j)) \geq -\beta; \\ G'(\varphi_j) \in \mathrm{L}((\alpha-\beta)Q) & \Longrightarrow \quad \mathrm{val}_Q(G'(\varphi_j)) \leq \alpha - \beta; \end{cases}$$

hence

$$\mathrm{val}_Q\left(\frac{G(\varphi_j)}{G'(\varphi_j)}\right) \geq -\alpha;$$

so

$$\frac{G(\varphi_j)}{G'(\varphi_j)} \in \mathrm{L}(\alpha Q).$$

Finally, we can deduce that $\varphi_{j+1} \in \mathrm{L}(\alpha Q)$.

As f and φ_N are in $\mathrm{L}(\alpha Q)$, $f - \varphi_N \in \mathrm{L}(\alpha Q)$. We recall that if $\psi \in \mathrm{L}(\alpha Q)$, if $\psi \neq 0$, the number of zeroes of ψ equals the number of poles of ψ (here α). Applied to $\psi = f - \varphi_N$, if $f \neq \varphi_N$, $\mathrm{val}_{P_i}(f-\varphi_N) \leq \alpha$ which is impossible from the construction of φ. It results that $f = \varphi_N$.

To check if $f \in \mathrm{B}^\star_\tau(y)$, one just have to count if $S = \{i \in \{1, \ldots, n\} / \ f(P_i) = y_i\}$ has at least $n - \tau$ elements; if it is the case, as we know $f \in \mathrm{L}(\alpha Q)$, *i.e.* $f \in L$, and $G(f) = 0$ from Theorem 2.

5.2 A tip for implementation: avoid doubles

Once we have found $f \in K$ such that $(Y - f)$ divides $G(Y)$, we can remove from our investigation all indexes $i \in I$ such that $f(P_i) = y_i$. This comes from the following proposition, which results directly from Proposition 7.

Proposition 8 *If*

$$\{i \in I/\ f_1(P_i) = y_i\} \cap \{i \in I/\ f_2(P_i) = y_i\} \neq \varnothing$$

then

$$\{i \in I/\ f_1(P_i) = y_i\} = \{i \in I/\ f_2(P_i) = y_i\}$$

and $f_1 = f_2$.

So if we have found φ such that $\varphi(P_j) = y_j$ for at least $n - \tau$ values of j, we can add the line:

"remove from I all $j \in \{1, \ldots, n\}$ such that $\varphi(P_j) = y_j$"

because the algorithm would return the same φ on such P_j's.

5.3 Complexity

Proposition 9 *Algorithm 2 takes* $\mathrm{O}(n^{\frac{3}{2}} \log \alpha / \sqrt{\alpha})$ *arithmetic operations in* K.

Proof: In algorithm 2, we know from Theorem 2 that $\deg G = b = \left\lfloor \frac{\lceil \sqrt{2\alpha n} \rceil}{\alpha} \right\rfloor$ In each **for** loop, we can evaluate $G(\varphi)$ and $G(\varphi)$ in $\mathrm{O}(b)$ with HÖRNER's rule [Knu98, p. 496]. We have $N = \log(\alpha + 1)$ and in the worst case, we have for each $i \in I$ a different solution, and $|I| = n$, hence we will loop n times.

6 Conclusion

We have improved steps 2 and 3 of SUDAN's algorithm and shown in proposition 5 that they can be done in $\mathrm{O}(n^2 k)$ arithmetic operations in $\mathbf{F}_q$. We proved in section 4 that τ-reconstruction can be done in $\mathrm{O}(n^4/k)$ arithmetic operations on $\mathbf{F}_q$ and that complexity is upper-bounded by the complexity of step 1 of SUDAN's algorithm. We also accelerate in section 5 steps 2 and 3 of SHOKROLLAHI and WASSERMAN generalization of SUDAN's algorithm to algebraic-geometric codes.

In both cases, the bottleneck seems to find polynomials that realize Theorem 1 (respectively Theorem 2).

Anyway, due to the condition $\mathrm{wdeg}_{1,(k-1)} Q(X, Y) < n - \tau$, SUDAN's algorithm is intrinsically useless for an information rate $R = \frac{k}{n} \geq \frac{1}{3}$ in its original version. However a new version of it is proposed in [GS98] that can be applied for any transmission rate. Note that in this paper, steps 2 and 3 remain unchanged in their article.

References

[BCS96] P. BÜRGISSER, M. CLAUSEN and M. A. SHOKROLLAHI. *Algebraic Complexity Theory*, volume 315 of *Grunlehren der Mathematischen Wissenschaften*. Springer-Verlag, 1996.

[CLR94] T. CORMEN, C. LEISERSON and R. RIVEST. *Introduction à l'algorithmique*. Dunod, Paris, 1994.

[Duu98] Iwan DUURSMA. Interpolation and approximation in decoding. In *IEEE 1998 Information Theory Workshop*, page 37, june 1998.

[Eli57] P. ELIAS. *List decoding for noisy channels*. Research Laboratory of Electronics, Massachusetts Institute of Technology, Cambridge, Mass., Rep. No. 335, 1957.

[GS98] V. GURUSWAMI and M. SUDAN. Improved decoding of Reed-Solomon codes and algebraic-geometric codes. preliminary version, communicated by Tom HØHOLDT, 1998.

[Knu98] D. E. KNUTH. *The Art of Computer Programming*, volume 2. Addison-Wesley, 3rd edition, 1998.

[Lan95] S. LANG. *Algebra*. Addison-Wesley, 3rd edition, 1995.

[MS88] F. J. MACWILLIAMS and N. J. A. SLOANE. *The Theory of Error-Correcting codes*. North-Holland Mathematical Library. North-Holland, 1988.

[Ret75] C. T. RETTER. Decoding Goppa codes with a BCH decoder. *IEEE Transactions on Information Theory*, 21:112, 1975.

[Sed89] R. SEDGEWICK. *Algorithms*. Addison-Wesley, 2nd edition, 1989.

[Sti93] H. STICHTENOTH. *Algebraic function fields and codes*. Springer-Verlag, 1993.

[Sud97a] M. SUDAN. Decoding of Reed-Solomon codes beyond the error-correction bound. *Journal of Complexity*, 13:180–193, 1997.

[Sud97b] M. SUDAN. Decoding Reed-Solomon codes beyond the error-correction diameter. In *35th Annual Allerton Conference on Communication, Control and Computing*, 1997.

[SW97] M. A. SHOKROLLAHI and H. WASSERMAN. Decoding algebraic-geometric codes beyond the error-correction bound. preprint, 1997.

[SW98] M. A. SHOKROLLAHI and H. WASSERMAN. Decoding algebraic geometric codes. In *IEEE 1998 Information Theory Workshop*, pages 40–41, June 1998.

[TS91] M. A. TSFASMAN and S. G. VLĂDUŢ. *Algebraic-Geometric Codes*. Mathematics and its Applications. Kluwer Academic Publishers, 1991.

[Zip93] R. ZIPPEL. *Effective Polynomial Computation*. Kluwer Academic Publishers, 1993.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399