List-Decoding of Binary Goppa Codes up to the Binary Johnson Bound

Daniel Augot, Morgan Barbier
INRIA Saclay Île-de-France
CNRS LIX, UMR 7161
École Polytechnique
91128 Palaiseau Cedex, France
Daniel.Augot@inria.fr
Morgan.Barbier@lix.polytechnique.fr

Alain Couvreur
Institut de Mathématiques de Bordeaux
Université Bordeaux I
351, Cours de la Libération
33405 Talence Cedex, France
Alain.Couvreur@math.u-bordeaux1.fr

***Abstract*—We study the list-decoding problem of alternant codes (which includes obviously that of classical Goppa codes). The major consideration here is to take into account the (small) size of the alphabet. This amounts to comparing the *generic* Johnson bound to the q-*ary* Johnson bound. The most favourable case is $q = 2$, for which the decoding radius is greatly improved.**

Even though the announced result, which is the list-decoding radius of binary Goppa codes, is new, we acknowledge that it can be made up from separate previous sources, which may be a little bit unknown, and where the binary Goppa codes has apparently not been thought at. Only D. J. Bernstein has treated the case of binary Goppa codes in a preprint. References are given in the introduction.

We propose an autonomous and simplified treatment and also a complexity analysis of the studied algorithm, which is quadratic in the blocklength n, when decoding ϵ-away of the relative maximum decoding radius.

I. Introduction

In 1997, Sudan presented the first list-decoding algorithm for Reed-Solomon codes [1]. Since the correction radius of Sudan's algorithm is larger, this represented an important milestone in algorithmic list-decoding Afterwards, Guruswami and Sudan [2] improved the previous algorithm by adding the notion of multiplicities. The number of errors that this algorithm is able to list-decode corresponds to the *Johnson radius* $e_\infty(n,d) = \left\lceil n - \sqrt{n(n-d)} \right\rceil - 1$, where d is the minimum distance of the code.

But, when the size q of the alphabet is properly taken into account, this radius is improved up to

$$e_q(n,d) = \left\lceil \theta_q \left(n - \sqrt{n\left(n - \frac{d}{\theta_q}\right)} \right) \right\rceil - 1, \qquad (1)$$

where $\theta_q = 1 - \frac{1}{q}$. See [3, Chap. 3] for a discussion about these kind of bounds. Dividing by n, and denoting $\delta = \frac{d}{n}$, we define $\tau_\infty(\delta) = \frac{e_\infty(n,d)}{n}$, and $\tau_q(\delta) = \frac{e_q(n,d)}{n}$, which are then equal to

$$\tau_\infty(\delta) = 1 - \sqrt{1-\delta}, \quad \tau_q(\delta) = \theta_q \left(1 - \sqrt{1 - \frac{\delta}{\theta_q}} \right) \qquad (2)$$

Note that $\tau_q(\delta)$ gets decreasingly close to $\tau_\infty(\delta)$ when q grows, and that $\tau_2(n,q)$ is the largest, see Fig. 1. We call $\tau_\infty(\delta)$ the *generic Johnson bound*, which does not take into account the size of the field, and $\tau_q(\delta)$ the *q-ary Johnson bound*. We present how to reach the $\tau_q(\delta)$ radius for the whole

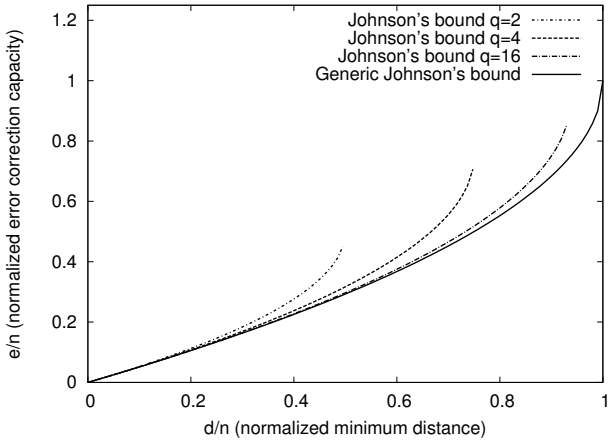

Fig. 1. The generic and q-ary *Johnson bound* $\tau_\infty(\delta)$ and $\tau_q(\delta)$

class of alternant codes. We have essentially compiled existing, but not very well-known results.

First, let us present the history of the results. Koetter and Vardy proposed in 2001, an "algebraic soft-decision" decoding algorithm for Reed-Solomon codes [4]. This method is based on an interpolation procedure, similar to Guruswami-Sudan's algorithm, with a two dimensional set of varying multiplicities, reflecting the reliability information given by the channel. Note that the idea of varying multiplicities was also considered in [2], as the "weighted polynomial reconstruction problem", but was not instantiated for particular cases. Before [4] circulated a preprint of Koetter and Vardy [5], which was a greatly extended version of [4], with many possible interesting instances of the weighted interpolation. In particular, the authors discussed the decoding of BCH codes over the binary symmetric channel, and reached in fact an error level which

978-1-4577-0437-6/11/$26.00 ©2011 IEEE

is nothing else than $\tau_2(\delta)$. Note that BCH codes are the most simple alternant codes.

Guruswami-Sudan's algorithm is in fact very general and applies to (one point) Algebraic Geometric codes [2]. By this manner, one also reaches the Johnson radius $e_\infty(n, d^\star)$, where $d^\star$ is the Goppa designed distance. Since it is possible, for a fixed alphabet $\mathbb{F}_q$, to construct Algebraic Geometric codes of any length, it makes sense to try to reach the q-ary Johnson bound $\tau_q(\delta)$, which was done in Guruswami's thesis [6].

Independently, Roth and Tal considered the q-ary list-decoding problem in [7], which is only a one page. Roth in his book [8] considers the list-decoding of alternant codes, and shows how to reach the q-ary Johnson radius $\tau_q(\delta)$, where δ is the minimum distance of the Generalised Reed-Solomon code from which the alternant code is built. Alternant codes were considered in [2], but only the generic Johnson Bound $\tau_\infty(\delta)$ was discussed there.

Among the alternant codes, the *binary Goppa codes* are particularly important, since they are used in the McEliece cryptosystem. They are not to be confused with Goppa's Algebraic Geometric codes, although there is a connection recalled in [9]. These codes are constructed with a *Goppa polynomial* G of degree r, and if G is square-free, then the distance of these codes is at least $2r + 1$ which is almost the double of $r+1$, which is what would be expected for a generic alternant code. In fact, using the fact that a Goppa code built with a square-free G is the same as the Goppa code built with G^2, the code can be decoded up to the radius

$$\left\lceil \frac{1}{2}\left(n - \sqrt{n\,(n-(4r+2))}\right)\right\rceil - 1. \tag{3}$$

But actually, the first author who really considered list-decoding of binary Goppa codes is D. J. Bernstein [10], in a preprint. His approach is different from interpolation based list-decoding algorithms. He starts with Patterson's algorithm [11] for decoding classical Goppa codes, and then reuses the information obtained by an unsuccessful application of Patterson's algorithm. It seems to be close to Wu's approach [12] for list-decoding Reed-Solomon and BCH codes. Wu can also reach the binary Johnson bound $\tau_2(\delta)$ for decoding *binary* BCH codes, while Bernstein's method decodes only up to the generic Johnson radius, apparently being not aware of the binary case.

This article is organised as follows. In Section II is recalled the list-decoding problem, the Johnson bounds, Section III gives the definitions of alternant codes and classical Goppa codes. Section IV deals with the decoding of alternant codes up to the q-ary Johnson bound.

II. List-decoding

Problem 1. The *list-decoding* problem of a code $\mathcal{C} \subset \mathbb{F}_q^n$ up to the radius $e \in [0, n]$ consists, for any y in $\mathbb{F}_q^n$, in finding *all* the codewords c in $\mathcal{C}$ such that $d(c, y) \leq e$.

The issue is to determine how large can e be such that the list is small. A partial answer is given by the so-called Johnson bound.

Theorem II.1. *Let C be a code of length n and minimum distance d over a alphabet of size q. Then any ball of radius e contains at most*

$$\frac{\frac{nd}{\theta_q}}{(n-\frac{e}{\theta_q})^2 - n(n-\frac{d}{\theta_q})}$$

codewords, provided the denominator is positive. The largest e such that this denominator is positive is called the Johnson radius*, and is equal to the $e_q(n, d)$ given in (1).*

For any given $\epsilon > 0$, if $e \leq (1-\epsilon)e_q(n, d)$, the size of the list is constant, for growing n, and is $O(n^2)$, for $e = e_q(n, d)$.

III. Classical Goppa Codes

This section is devoted to the presentation of classical q–ary Goppa codes, as alternant codes. Let q be an arbitrary prime power and m, n be two integers such that $m \geq 2$ and $n \leq q^m$. Let also $L \triangleq (\alpha_1, \ldots, \alpha_n)$ denote n distinct elements of $\mathbb{F}_{q^m}$.

Definition III.1. Let r be an integer such that $0 < r < n$. Let $G \in \mathbb{F}_{q^m}[X]$ be a polynomial of degree r which does not vanish at any element of L. The q–ary classical Goppa code $\Gamma_q(L, G)$ is defined by

$$\Gamma_q(L, G) \triangleq \left\{ \mathbf{c} \in \mathbb{F}_q^n \;\middle|\; \sum_{i=1}^{n} \frac{c_i}{X - \alpha_i} \equiv 0 \mod G \right\}.$$

Classical Goppa codes are contained in the larger class of *alternant codes* [13, Chap 12] which are subfield subcodes of Generalised Reed–Solomon Codes.

Definition III.2 (Generalised Reed–Solomon code)**.** Let $B \triangleq (\beta_1, \ldots, \beta_n)$ be an n–tuple of elements of $\mathbb{F}_{q^m}^*$. Let k be a positive integer. The *Generalised Reed–Solomon* code (or GRS code) over $\mathbb{F}_{q^m}$ associated to the triple (L, B, k) is the image of the evaluation map:

$$\mathrm{ev}_{L,B} : \begin{cases} \mathbb{F}_{q^m}[X]_{<k} & \to & \mathbb{F}_{q^m}^n \\ f(X) & \mapsto & (\beta_1 f(\alpha_1), \ldots, \beta_n f(\alpha_n)) \end{cases}.$$

The following proposition describes Goppa codes as alternant codes. Its proof can be found in [13, Chap. 12].

Proposition III.1. *Let r be an integer such that $0 < r < n$ and $G \in \mathbb{F}_{q^m}[X]$ be a polynomial of degree r which does not vanish at any element of L. Then, the classical Goppa code $\Gamma_q(L, G)$ is the subfield subcode $GRS_{q^m}(L, B, n-r)_{|\mathbb{F}_q}$, where $B = (\beta_1, \ldots, \beta_n)$ is defined by*

$$\forall i \in \{1, \ldots, n\},\ \beta_i \triangleq \frac{G(\alpha_i)}{\prod_{j \neq i}(\alpha_i - \alpha_j)}.$$

When the Goppa polynomial G is well chosen, then the minimum distance is better than expected, see [14], [15].

Theorem III.1. *Let $G \in \mathbb{F}_{q^m}[X]$ be square-free polynomial which does not vanish at any element of L and such that $0 < \deg(G) < n/q$. Then, $\Gamma_q(L, G^{q-1}) = \Gamma_q(L, G^q)$ and this code has parameters $[n, \geq n - m(q-1)\deg G, \geq q \deg G + 1]_q$.*

IV. List decoding of classical Goppa codes as evaluation codes

A. List-decoding of general alternant codes

We give treatment of the list-decoding algorithm for alternant codes, up to the q-ary Johnson bound. Clearly, this algorithm can be obtained quite easily from [6] and [5]. However, this presentation has the advantages to be simple and self-contained. In addition, we give a detailed complexity analysis. As far as we know, such analysis has never been written down up to now.

Definition IV.1. Let $Q(X,Y) = \sum_{i,j} Q_{ij} X^i Y^j \in \mathbb{F}_{q^m}[X,Y]$ be a bivariate polynomial and $u \geq 0$. We say that $Q(X,Y)$ has *multiplicity at least* u *at* $(0,0)$ if $Q_{ij} = 0$ for all (i,j) such that $i+j < u$. We say that Q has *multiplicity at least* u *at point* $(a,b) \in \mathbb{F}_{q^m}^2$ if $Q(X+a, Y+b)$ has multiplicity at least u at $(0,0)$. We denote this fact by $\mathrm{mult}\,(Q(X,Y),(a,b)) \geq s$.

Definition IV.2. For $u, v \in \mathbb{N}$, the weighted degree $\mathrm{wdeg}_{u,v}(Q(X,Y))$ of a polynomial $Q(X,Y) = \sum Q_{ij} X^i Y^j$ is $\max\{ui + vj,\ (i,j) \in \mathbb{N} \times \mathbb{N}\ |Q_{ij} \neq 0\}$.

Consider a $[n, k_{GRS}, d_{GRS}]_{q^m}$ GRS(L, B, k_{GRS}) code, and the corresponding alternant code $\mathcal{C} \triangleq GRS_{|\mathbb{F}_q}$. We list-decode up to γn errors, where γ is to be determined further.

Let $y \in \mathbb{F}_q^n$ be a received word. The main steps of the algorithm are the following: Interpolation, Root-Finding, Reconstruction. An auxiliary $s \in \mathbb{N} \setminus \{0\}$ is needed, which is also discussed further. Now we can sketch the algorithm.

1) Interpolation: Find $Q(X,Y) = \sum Q_i(X) Y^i \in \mathbb{F}_{q^m}[X,Y]$ such that
   a) (non triviality) $Q(X,Y) \neq 0$;
   b) (interpolation with varying multiplicities)
      - $\mathrm{mult}(Q(X,Y),(\alpha_i, y_i\beta_i^{-1})) \geq s(1-\gamma)$;
      - $\mathrm{mult}(Q(X,Y),(\alpha_i, z\beta_i^{-1})) \geq \frac{s\gamma}{q-1}$, for any $z \in \mathbb{F}_q \setminus \{y_i\}$;
   c) (weighted degree) $\mathrm{wdeg}_{1,k_{GRS}}\, Q(X,Y) < sn\left((1-\gamma)^2 + \frac{\gamma^2}{q-1}\right)$;
2) Root-Finding: Find all the factors $(Y - f(X))$ of $Q(X,Y)$, with $\deg f(X) < k_{GRS}$;
3) Reconstruction: Compute the codewords associated to the $f(X)$'s found in the Root-Finding step, using the evaluation map $\mathrm{ev}_{L,B}$. Retain only those which are at distance at most γn from y.

From [2], we have:

Lemma IV.1. *Let u be an integer and $Q(X,Y)$ be a polynomial with multiplicity u at (a,b). Then, for any $f(X)$ such that $f(a) = b$, one has $(X-a)^u \mid Q(X, f(X))$.*

Now we can state the correctness of the algorithm.

Proposition IV.1. *Let y be the received word, and $Q(X,Y)$ satisfying conditions 1a, 1b, and 1c above. Let $f(X)$ be a polynomial such that $\deg f(X) < k_{GRS}$ and accordingly, let $c = \mathrm{ev}_{L,B}(f(X))$. If $d(c,y) \leq \gamma n$, then $Q(X, f(X)) = 0$.*

Proof: Assume that $d(\mathrm{ev}_{L,B}(f(X)), y) = \kappa n \leq \gamma n$ and $Q(X, f(X)) = 0$. Set $I \triangleq \left\{i, f(\alpha_i) = y_i\beta_i^{-1}\right\}$ and $\overline{I} \triangleq \left\{i, f(\alpha_i) \neq y_i\beta_i^{-1}\right\}$. Obviously we have $|I| = n(1-\kappa)$ and $|\overline{I}| = \kappa n$. Note that, from Lemma IV.1, $Q(X, f(X))$ is divisible by

$$\prod_{i \in I} (X - \alpha_i)^{\lceil s(1-\gamma)\rceil} \times \prod_{j \in \overline{I}} (X - \alpha_j)^{\lceil s\gamma/(q-1)\rceil},$$

which has degree $D = n(1-\kappa)\lceil s(1-\gamma)\rceil + n\kappa\left\lceil \frac{s\gamma}{q-1}\right\rceil$. This degree is a decreasing function of κ for $\gamma < \frac{q-1}{q}$, since it is an affine function of the variable κ, whose leading term is

$$\kappa n\left(\left\lceil \frac{s\gamma}{q-1}\right\rceil - \lceil s(1-\gamma)\rceil\right) < 0.$$

The minimum is reached for $\kappa = \gamma$, and hence is greater than $sn\left((1-\gamma)^2 + \frac{\gamma^2}{q-1}\right)$. Thus, $D \geq sn\left((1-\gamma)^2 + \frac{\gamma^2}{q-1}\right)$. On the other hand, the weighted degree condition imposed $Q(X,Y)$ implies that $\deg Q(X, f(X)) < sn\left((1-\gamma)^2 + \frac{\gamma^2}{q-1})\right)$. Since $Q(X, f(X))$ has more roots than its degree, it is zero. ∎

Proposition IV.2. *Let $\delta_{GRS} = \frac{d_{GRS}}{n}$ be the relative minimum distance of a GRS code as above, defining an alternant code over $\mathbb{F}_q$. Let $\tau_q(\delta_{GRS})$ be as in (2). Then, for any $\gamma < \tau$, there exists s large enough such that a polynomial $Q(X,Y)$, satisfying the three previous constraints 1a, 1b, and 1c, always exists, whatever the received word $y \in \mathbb{F}_q^n$ is.*

Proof: To make sure that, for every received word y, a non zero $Q(X,Y)$ exists, it is enough to find $\tau_q(\delta_{GRS})$ such that we have more indeterminates than equations in the linear system given by 1b, and 1c. This leads to the following inequality after simple computations (see [9, App. A & B] for further details):

$$\frac{s^2n^2((1-\gamma)^2 + \frac{\gamma^2}{q-1})^2}{2(k-1)} > \left(\binom{s(1-\gamma)+1}{2} + (q-1)\binom{s\frac{\gamma}{q-1}+1}{2}\right)n, \quad (4)$$

which can be rewritten as

$$\left((1-\gamma)^2 + \frac{\gamma^2}{q-1}\right)^2 > R'\left((1-\gamma)^2 + \frac{\gamma^2}{q-1} + \frac{1}{s}\right),$$

where $R' \triangleq \frac{k_{GRS}-1}{n}$. Thus, we find that $\mu \triangleq \mu(\gamma) \triangleq (1-\gamma)^2 + \frac{\gamma^2}{q-1}$ must satisfy $\mu^2 - R'\mu - \frac{R'}{s} > 0$. The roots of the equation $\mu^2 - R'\mu - \frac{R'}{s} = 0$ are

$$\mu_0 = \frac{R' - \sqrt{R'^2 + 4\frac{R'}{s}}}{2}, \quad \mu_1 = \frac{R' + \sqrt{R'^2 + 4\frac{R'}{s}}}{2}.$$

Note that the function $\mu(\gamma)$ is decreasing for $\gamma \in [0, 1 - \frac{1}{q}]$. Only μ_1 is positive and thus we must have $\mu > \mu_1$, i.e.

$$(1-\gamma)^2 + \frac{\gamma^2}{q-1} > \mu_1. \quad (5)$$

Again, we have two roots for the equation $(1-\gamma)^2+\frac{\gamma^2}{q-1}=\mu_1$:

$$\gamma_0 = \frac{q-1}{q}\left(1-\sqrt{1-\frac{q}{q-1}(1-\mu_1)}\right) \quad (6)$$

$$\gamma_1 = \frac{q-1}{q}\left((1+\sqrt{1+\frac{q}{q-1}(1-\mu_1)}\right). \quad (7)$$

Only $\gamma_0 < \frac{q-1}{q}$, and thus we must have

$$\gamma < \gamma_0 = \frac{q-1}{q}(1-\sqrt{1-\frac{q}{q-1}(1-\mu_1)}. \quad (8)$$

Then, when $s \to \infty$, we have $\mu_1 \to R'$, and we get

$$\gamma < \tau = \frac{q-1}{q}\left(1-\sqrt{1-\frac{q}{q-1}(1-R')}\right) \quad (9)$$

Using the fact that $k_{GRS}-1 = n-d_{GRS}$, i.e $R' = 1-\delta_{GRS}$, we get

$$\tau = \frac{q-1}{q}\left(1-\sqrt{1-\frac{q}{q-1}\delta_{GRS}}\right),$$

which is exactly the q-ary Johnson radius. ∎

The above bound is better than the error correction capacities of the previous algorithms [2], [10]. For the binary case, we plot in Figure 2 the binary Johnson bound, the generic Johnson bound, and the unambiguous decoding bound $t = \frac{d-1}{n}$, or asymptotically $\delta/2$. As usually, the higher the normalised minimum distance is, the better the Johnson bound is.

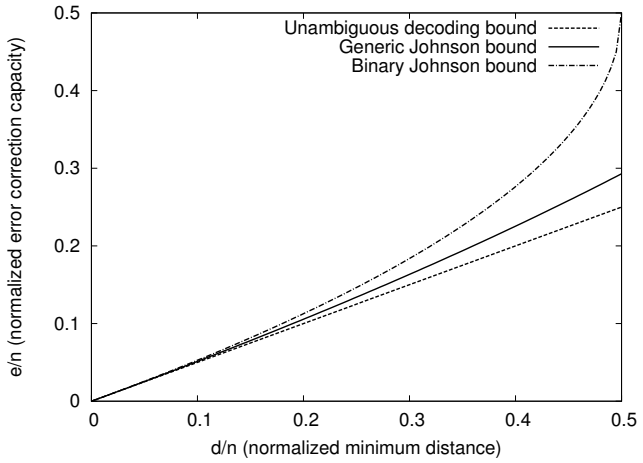

Fig. 2. Relative decoding radii for some algorithms for binary alternant codes, including binary square-free Goppa codes.

B. Complexity Analysis

The main issue is to find how large the order of multiplicity s has to be, to approach closely the limit correction radius $\tau(\delta_{GRS})$ given by (2).

Lemma IV.2. *To list-decode up to a relative radius of $\gamma = (1-\varepsilon)\tau$, it is enough to have an auxiliary multiplicity s of size $\mathcal{O}(\frac{1}{\varepsilon})$, where the constant in the big-$\mathcal{O}$ depends only on q and on the pseudo-rate $R' = \frac{k_{GRS}-1}{n}$ of the GRS code.*

Proof: To get the dependency on s, we work out Equation (8). Let us denote by $\gamma(s)$ the achievable relative correction radius for a given s. We have

$$\gamma(s) = \frac{q-1}{q}\left(1-\sqrt{1-\frac{q}{q-1}(1-\mu_1(s))}\right), \quad (10)$$

with $\mu_1(s) = \frac{R'+\sqrt{R'^2+4\frac{R'}{s}}}{2}$. We use repeatedly that $\sqrt{1+x} < 1+\frac{x}{2}$, for all $x>0$. First, we have the bound:

$$\mu_1(s) = \frac{R'+\sqrt{R'^2+4\frac{R'}{s}}}{2} \quad (11)$$

$$= \frac{R'}{2}\left(1+\sqrt{1+\frac{4}{sR'}}\right) \quad (12)$$

$$\leq \frac{R'}{2}\left(1+\left(1+\frac{4}{2sR'}\right)\right) \quad (13)$$

$$= R'+\frac{1}{s}. \quad (14)$$

Now, calling $K_q(R')$ the quantity $1-\frac{q}{q-1}(1-R')$:

$$\gamma(s) = \frac{q-1}{q}\left(1-\sqrt{1-\frac{q}{q-1}(1-\mu_1(s))}\right) \quad (15)$$

$$\geq \frac{q-1}{q}\left(1-\sqrt{1-\frac{q}{q-1}\left(1-R'-\frac{1}{s}\right)}\right) \quad (16)$$

$$= \frac{q-1}{q}\left(1-\sqrt{K_q(R')+\frac{q}{q-1}\frac{1}{s}}\right) \quad (17)$$

$$= \frac{q-1}{q}\Bigg(1- \sqrt{K_q(R')\left(1+\frac{q}{q-1}\frac{1}{s}\frac{1}{K_q(R')}\right)}\Bigg) \quad (18)$$

$$\geq \frac{q-1}{q}\Bigg(1-\sqrt{K_q(R')} -\sqrt{K_q(R')}\frac{1}{2}\frac{q}{q-1}\frac{1}{s}\frac{1}{K_q(R')}\Bigg) \quad (19)$$

$$= \frac{q-1}{q}\left(1-\sqrt{K_q(R')}\right)-\frac{1}{2}\frac{1}{s\sqrt{K_q(R')}} \quad (20)$$

$$= \tau_q(\delta)-\frac{1}{2s\sqrt{K_q(R')}} \quad (21)$$

$$= \tau_q(\delta)\left(1-\frac{1}{2s\tau_q(\delta)\sqrt{K_q(R')}}\right) \quad (22)$$

Thus, to reach a relative radius $\gamma = (1-\varepsilon)\tau$, it is enough to take

$$s = \frac{1}{2\varepsilon\tau\sqrt{K_q(R')}} = \mathcal{O}(\frac{1}{\varepsilon}). \quad (23)$$

∎

The most expensive computational step in these kinds of list-decoding algorithms is the interpolation step, while root-finding is cheaper [16], [17]. We focus on the complexity of this step. We assume the use of the so-called Koetter

algorithm [18] (see for instance [19], [20] for a recent exposition), to estimate the complexity. This algorithm has complexity $\mathcal{O}(lC^2)$, where l is the Y-degree of the $Q(X,Y)$ polynomial, and C is the number of linear equations given by the interpolation constraints.

Corollary IV.1. *The proposed list-decoding runs in $\mathcal{O}(\frac{1}{\varepsilon^5}n^2)$ field operations to list-decode up to $(1-\varepsilon)\tau_q(\delta)\cdot n$ errors, where the constant in the big-$\mathcal{O}$ depends only on q and the pseudo-rate R'.*

Proof: Assume that we would like to decode up to $n\gamma = n(1-\varepsilon)\tau_q(\delta)$. The number of equations given by the interpolation conditions can be seen to be $\mathcal{O}(ns^2)$ (see Equation (4)). Now, the list size ℓ is bounded above by the Y-degree of the interpolation polynomial $Q(X,Y)$, which is at most

$$\frac{sn((1-\gamma)^2+\frac{\gamma^2}{q-1})}{k-1} = \mathcal{O}(s), \tag{24}$$

for fixed $R' = \frac{k-1}{n}$. Fitting $s = \mathcal{O}(\frac{1}{\varepsilon})$ (see (23)), we conclude that this method runs in $\mathcal{O}(n^2\varepsilon^{-5})$. For the Root-Finding step, in [17], an algorithm of complexity $\mathcal{O}(\ell^3k^2)$ is proposed, assuming q is small, and we get $\mathcal{O}(s^3n^2)$, which is less than the cost of the interpolation step. ∎

Corollary IV.2. *To reach the non relative q-ary Johnson radius $e_q(n,d)$ given in Eq. 1, it is enough to have $s = \mathcal{O}(n)$. The complexity is then $\mathcal{O}(n^7)$, where the constant in the big-$\mathcal{O}$ only depends on q and R'.*

Proof: It is enough to consider Lemma IV.2 and the fact that

$$e_q(n,d) = n\tau_q(\delta)(1-\varepsilon),$$

with $\varepsilon = \mathcal{O}(\frac{1}{n})$. ∎

C. Application to classical binary Goppa codes

The application of this algorithm to binary Goppa codes defined with a square-free polynomial G uses Theorem III.1 which yields

$$\Gamma_2(L,G) = \Gamma_2(L,G^2).$$

This code benefits from the dimension of $\Gamma_2(L,G)$ and from the distance of $\Gamma_2(L,G^2)$. Thus, if $\deg G = t$, then we have a decoding radius of $\left\lceil \frac{1}{2}\left(n-\sqrt{n(n-4t-2)}\right)\right\rceil - 1$. Note also the q-ary Goppa codes can also be decoded up to the q-ary Johnson radius, and taking into account the improvement of the distance given by Theorem III.1.

An important note is that the distance given in the achieved decoding radius correspond to the Goppa designed distance $d^\star = 2r+1$. Classical Goppa codes lie on the Gilbert-Varshamov bound, and have better true minimum distance, but it is out of algorithmic reach. Also, when speaking about asymptotic figures, one must keep in mind that, for keeping the relative designed distance $\delta = d^\star/n$ constant, the rate has to (logarithmically) go to zero with the length.

Finally, we have not dealt with so called fast arithmetic, where elementary fast bricks from computer algebra can be used. Let us remark that after [9], Bernstein published [21] on this topic. Previous authors already considered such fast methods [22], [23] for Reed-Solomon codes and several AG (Hermitian and Trace-Norm) codes [24].

REFERENCES

[1] M. Sudan, "Decoding of Reed-Solomon codes beyond the error-correction bound," *Journal of Complexity*, vol. 13, no. 1, pp. 180 – 193, 1997.

[2] V. Guruswami and M. Sudan, "Improved decoding of Reed-Solomon and algebraic-geometry codes," *Information Theory, IEEE Transactions on*, vol. 45, no. 6, pp. 1757 –1767, 1999.

[3] V. Guruswami, *Algorithmic results in list decoding*. Now Publishers Inc, 2007.

[4] R. Kötter and A. Vardy, "Algebraic soft-decision decoding of Reed-Solomon codes," *Information Theory, IEEE Transactions on*, vol. 49, no. 11, pp. 2809–2825, 2003.

[5] A. Vardy and R. Koetter, "Algebraic Soft-Decision Decoding of Reed-Solomon Codes," 2000, 53 pages, circulated before the IEEE publication.

[6] V. Guruswami, *List Decoding of Error-Correcting Codes - Winning Thesis of the 2002 ACM Doctoral Dissertation Competition*, ser. Lectures Notes in Computer Science. Springer, 2004, vol. 3282.

[7] I. Tal and R. M. Roth, "On list decoding of alternant codes in the Hamming and Lee metrics," in *Information Theory, 2003. Proceedings. IEEE International Symposium on*, 2003, p. 364.

[8] R. Roth, *Introduction to Coding Theory*. Cambridge University Press, 2006.

[9] D. Augot, M. Barbier, and A. Couvreur, "List-decoding of binary Goppa codes up to the binary Johnson bound," INRIA, Research Report RR-7490, 12 2010. [Online]. Available: http://hal.inria.fr/inria-00547106/en/

[10] D. J. Bernstein, "List decoding for binary Goppa codes," 2008, http://cr.yp.to/papers.html#goppalist.

[11] N. Patterson, "The algebraic decoding of Goppa codes," *Information Theory, IEEE Transactions on*, vol. 21, no. 2, pp. 203–207, 1975.

[12] Y. Wu, "New list decoding algorithms for Reed-Solomon and BCH codes," *Information Theory, IEEE Transactions on*, vol. 54, no. 8, pp. 3611–3630, 2008.

[13] F. J. Macwilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*, ser. North-Holland Mathematical Library. North Holland, 1983.

[14] H. Stichtenoth, *Algebraic function fields and codes*, 2nd ed., ser. Universitext. Berlin: Springer-Verlag, 2009.

[15] D. J. Bernstein, T. Lange, and C. Peters, "Wild McEliece," Cryptology ePrint Archive, Report 2010/410, 2010, http://eprint.iacr.org/.

[16] D. Augot and L. Pecquet, "A Hensel lifting to replace factorization in list decoding of algebraic-geometric and Reed-Solomon codes," *Information Theory, IEEE Transactions on*, vol. 46, no. 7, pp. 2605–2613, 2000.

[17] R. M. Roth and G. Ruckenstein, "Efficient decoding of Reed-Solomon codes beyond half the minimum distance," *Information Theory, IEEE Transactions on*, vol. 46, no. 1, pp. 246–257, 2000.

[18] R. Kötter, "On algebraic decoding of algebraic-geometric and cyclic codes," Ph.D. dissertation, University of Linköping, 1996.

[19] P. V. Trifonov, "Efficient Interpolation in the Guruswami-Sudan Algorithm," *Information Theory, IEEE Transactions on*, vol. 56, no. 9, pp. 4341–4349, 2010.

[20] R. Koetter, J. Ma, and A. Vardy, "The Re-Encoding Transformation in Algebraic List-Decoding of Reed-Solomon Codes," 2010. [Online]. Available: http://arxiv.org/abs/1005.5734

[21] D. J. Bernstein, "Simplied high-speed high-distance list decoding for alternant codes," 2011, http://cr.yp.to/papers.html#simplelist.

[22] M. Alekhnovich, "Linear diophantine equations over polynomials and soft decoding of Reed-Solomon codes," *Information Theory, IEEE Transactions on*, vol. 51, no. 7, pp. 2257 –2265, july 2005.

[23] P. Beelen and K. Brander, "Key equations for list decoding of Reed-Solomon codes and how to solve them," *Journal of Symbolic Computation*, vol. 45, no. 7, pp. 773–786, 2010.

[24] ——, "Efficient list decoding of a class of algebraic-geometry codes," *Advances in Mathematics of Communication*, vol. 4, no. 4, pp. 485–518, 2010.