AAECC (2013) 24:237–253
DOI 10.1007/s00200-013-0191-2

ORIGINAL PAPER

List decoding of repeated codes

Fernando Hernando · Michael O'Sullivan · Diego Ruano

Received: 20 October 2012 / Revised: 28 February 2013 / Accepted: 25 March 2013 /
Published online: 2 July 2013

Abstract Assuming that we have a soft-decision list decoding algorithm of a linear code, a new hard-decision list decoding algorithm of its repeated code is proposed in this article. Although repeated codes are not used for encoding data, due to their parameters, we show that they have a good performance with this algorithm. We compare, by computer simulations, our algorithm for the repeated code of a Reed–Solomon code against a decoding algorithm of a Reed–Solomon code. Finally, we estimate the decoding capability of the algorithm for Reed–Solomon codes and show that performance is somewhat better than our estimates.

This research was partially supported by the National Science Foundation under Grant No. CCF-0916492, by the Danish National Research Foundation and the National Science Foundation of China (Grant No.11061130539) for the Danish-Chinese Center for Applications of Algebraic Geometry in Coding Theory and Cryptography, by the Spanish grant No. MTM2007-64704, and by the Spanish MINECO under grant No. MTM2012-36917-C03-03.

F. Hernando
Department of Mathematics, Universidad Jaume I, Campus Riu Sec Castellón de la Plana, 12071, Spain
e-mail: carrillf@mat.uji.es

F. Hernando
Department of Mathematical Sciences, Aalborg University, Aalborg, Denmark

M. O'Sullivan
Department of Mathematics and Statistics, San Diego State University, San Diego, CA, 92182-7720, USA
e-mail: mosullivan@mail.sdsu.edu

D. Ruano (✉)
Department of Mathematical Sciences, Aalborg University,
Fr. Bajers Vej 7G, 9220 Aalborg Øst, Denmark
e-mail: diego@math.aau.dk

Keywords Linear codes · Matrix-product codes · Decoding algorithm · Minimum distance · Quasi-cyclic codes

Mathematics Subject Classification (2000) 94B05 · 94B35

1 Introduction

List decoding was introduced by Elias [4] and Wozencraft [12]. A list decoder can produce several candidate codewords near the received vector, thus relaxing the requirements of unique decoding and enabling the possibility of decoding beyond half of the minimum distance. The efficient list decoding problem was unsolved for many years until Sudan [11] provided an algorithm for low rate Reed–Solomon codes. Later Guruswami and Sudan [5] gave a general answer for Reed–Solomon codes that has 2 steps: the interpolation step and the root finding step.

A soft-decoding algorithm works under the assumption that the output of the channel is probabilistic information for the reliability of input data in contrast to a hard-decision list-decoding algorithm which assumes that a word of the ambient space is received. Koetter and Vardy [6] created a soft-decision list decoding algorithm for RS codes based on the interpolation techniques of Guruswami–Sudan. This algorithm has 3 steps. First, probabilistic information from the channel is translated into an assignment of multiplicities to points in the plane—the points representing received location-value pairs. The other two steps are the interpolation and root finding steps.

In this paper we consider the repeated code of a linear code $C \subset \mathbb{F}_q^n$, that is $C^\ell = \{(c, \ldots, c) : c \in C\} \subset \mathbb{F}_q^{\ell n}$, for some $\ell \geq 2$. We present a (hard-decision) list decoding algorithm for repeated codes, based on a soft-decision list decoding algorithm for the constituent code, C. In our algorithm, the multiplicity step is based on the algebraic structure of the code, rather than the information from the channel. Namely, it is an interpolation problem for a single block taking into account the information provided by the whole codeword. As far as the authors know, this is a novel idea in interpolation decoding. We consider two multiplicity assignment methods in detail, each of them gives rise to a different decoding algorithm. The first choice maximizes the score and, therefore, the error correction capability. The second one minimizes the sum of the multiplicities, thus it minimizes the computational time complexity.

Repeated codes do not have good parameters, but we remark that the minimum distance of a linear code is only an estimate of unique-decoding capability, and it is an even rougher estimate of the list-decoding performance of the code. We show that repeated codes of Reed–Solomon codes using the algorithm in [8] and taking into account our new set up for the multiplicities, may have similar decoding capability to a Reed–Solomon code, with the same information ratio and length, using the algorithm in [9]. However, significant differences in the computation time are observed since the complexity of the first simulation depends on n instead of ℓn, the length of the code. We shall compare our hard-decision list-decoding algorithm to another hard-decision list-decoding algorithm, since it does not make sense to compare it to a soft-decoding algorithm.

Finally, we estimate the decoding capability of the algorithm for Reed–Solomon codes. Even though the bounds assume certain properties of the error vector, they are relatively close to experimental values in the examples.

The paper is organized as follows: In Sect. 2 we recall the soft-decision interpolation problem for RS codes. In Sect. 3 we introduce our list decoding algorithm for repeated codes. In Sect. 4, we present some simulations of the decoding algorithm with MAGMA [2] and interpret the results. We estimate the number of errors t that we can decode in Sections 5 and 6. Section 7 concludes the article.

2 Soft-decoding

Koetter and Vardy [6] discovered a soft-decision list decoding algorithm for RS codes based on the interpolation techniques of Guruswami–Sudan, and later the interpolation step was described using Gröbner bases in [1,9]. For our simulations, we use the algorithms in [8,9] , which use the same approach, the only difference being the multiplicity assignment. We recall in this section the soft-decoding algorithm in [8] for Reed–Solomon codes.

Let $\alpha_1, \ldots, \alpha_n$ be n different points of the finite field $\mathbb{F}_q$ with q elements and let C be the Reed–Solomon code with parameters $[n, k, d]$ defined as

$$C = \{(h(\alpha_1), h(\alpha_2), \ldots, h(\alpha_n)) : \deg(h) \leq k - 1\}.$$

For soft decision decoding, Koetter and Vardy [6] use reliability information provided by the channel to assign multiplicities $m_{i,\beta}$ to each point $p_{i,\beta} = (\alpha_i, \beta)$ for $i = 1, \ldots, n$ and $\beta \in \mathbb{F}_q$. Let M be the collection of these multiplicities,

$$M = \{(p_{i,\beta}, m_{i,\beta}) : i = 1, \ldots, n; \beta \in \mathbb{F}_q\}$$

Consider the ideal in $\mathbb{F}_q[x, y]$ of polynomials interpolating at the points of M with the desired multiplicities:

$$I_M = \{f \in \mathbb{F}_q[x, y] : \text{mult}_p(f) \geq m \; for \; (p, m) \in M\}, \tag{1}$$

where $\text{mult}_p(f)$ denotes the multiplicity of f at p. For $r = (r_1, \ldots, r_n) \in \mathbb{F}_q^n$, let h_r be the interpolating polynomial at the points (α_i, r_i) for $i = 1, \ldots, n$. The key observation of Guruswami and Sudan is that for $f \in I_M$ and a codeword c, $y - h_c$ is a factor of $f(x, y)$ when

$$\sum_{i=1}^{n} m_{i,c_i} > \deg(f(x, h_c)) \tag{2}$$

For a given M we therefore define for each $r = (r_1, \ldots, r_n) \in \mathbb{F}_q^n$

$$\text{score}(r) = \sum_{i=1}^{n} m_{i,r_i}.$$

We use the $(1, k - 1)$-weighted degree of polynomial $f = \sum f_{i,j} x^i y^j$, which is defined to be $\deg_{1,k-1}(f) = \max\{i + j(k - 1) : f_{i,j} \neq 0\}$. Extend the weighted degree to a monomial ordering by taking $x^a y^b >_{k-1} x^i y^j$ if either $\deg_{1,k-1}(x^a y^b) > \deg_{1,k-1}(x^i y^j)$ or if $\deg_{k-1}(x^a y^b) = \deg_{k-1}(x^i y^j)$ and $b > j$. We may now interpret (2) as saying that a polynomial $f(x, y)$ such that the score of c is larger than the $(1, k-1)$-degree of $f(x, y)$ is divisible by $y - h_c$. Factoring f would produce the high scoring codeword c.

The requirement that $f(x, y)$ pass through (α_i, β) with multiplicity $m_{i,\beta}$ imposes $\binom{m_{i,\beta}+1}{2}$ conditions, so overall we have

$$N = \sum_{i=1}^{n} \sum_{\beta \in \mathbb{F}_q} \binom{m_{i,\beta}+1}{2}, \tag{3}$$

conditions. There is an upper bound for the $(1, k-1)$-weighted degree d of a polynomial $Q(x, y)$ given that N conditions are imposed (see e.g. [8, Proposition 3]).

Hence, we look for an interpolating polynomial of the form $\sum_{(i,j)\in S} f_{i,j} x^i y^j$, where $S \subset \{(i, j) : i, j \geq 0\}$ is the subset of indices with $(1, k-1)$-degree less than d, and by choice of d, $|S| > N$. The minimal polynomial with respect to the $>_{k-1}$ is usually called $Q(x, y)$. One method for obtaining $Q(x, y)$, for example [8], is to compute a Gröbner basis of I_M with respect to the $(1, k-1)$-weighted degree and pick the smallest element in it.

3 List decoding of repeated codes

Let $C \subset \mathbb{F}_q^n$ be a Reed–Solomon code with parameters $[n, k, d]$ and generator matrix G. We will describe the algorithm for a Reed–Solomon code, however, the algorithm can be extended in a straightforward manner to any linear code provided with soft-decision list-decoding algorithm.

We consider the repeated code of C,

$$C^\ell = \{(c, \ldots, c) : c \in C\},$$

which has parameters $[\ell n, k, \ell d]$ and generator matrix $(G|\cdots|G)$ [10, Problem 17 of Ch. 1]. We will describe a hard-decision list decoding algorithm for C^ℓ by using a soft-decoding algorithm for C, thus we only have to define the matrix of multiplicities from a received word.

Let $c = (c_1, \ldots, c_n) \in C$ then a typical codeword of C^ℓ is of the form

$$\mathbf{c} = (c_1, \ldots, c_n, \ldots, c_1, \ldots, c_n) \in \mathbb{F}_q^{\ell n}.$$

One can also understand a vector of length ℓn as an $\ell \times n$ matrix. Hence, if $\mathbf{v} \in \mathcal{M}(\ell \times n, \mathbb{F}_q)$ we denote by v_i^j the entry corresponding to the jth row (block) and the ith column (position), for $i = 1, \ldots, n$ and $j = 1, \ldots, \ell$. According to this notation a word in $\mathbb{F}_q^{\ell n}$ may be represented as

$$\mathbf{v} = (v_1^1, \ldots, v_n^1, \ldots, v_1^\ell, \ldots, v_n^\ell).$$

Let $\mathbf{c}$ be the sent word and $\mathbf{r} = \mathbf{c} + \mathbf{e}$ the received word with error weight $t = wt(\mathbf{e})$. We have that $\mathbf{r} = (r_1^1, \ldots, r_n^1, \ldots, r_1^\ell, \ldots, r_n^\ell)$ and $\mathbf{c} = (c_1^1, \ldots, c_n^1, \ldots, c_1^\ell, \ldots, c_n^\ell)$. Since $\mathbf{c} \in C^\ell$, one has that

$$c_i^j = c_i^k, \text{ for every } i \in \{1, \ldots, n\} \text{ and } j, \quad k \in \{1, \ldots, \ell\}.$$

Therefore, $r_i^j = r_i^k$ if and only if $e_i^j = e_i^k$. Hence if the received word has the same value in several positions it is likely that the error value in these positions is zero. Namely, the more positions where the values agree the more likely that these positions are error free. Moreover, the bigger the base field $\mathbb{F}_q$ the more likely the previous assumption is right. Based on this fact we will define the multiplicities $m_{i,\beta}$ at the point $p_{i,\beta} = (\alpha_i, \beta)$ for the soft-decoding algorithm of C.

For a received word $\mathbf{r}$, we define two different assignment of multiplicities. For $i = 1, \ldots, n$ and $\beta \in \mathbb{F}_q$:

(1) $m_{i,\beta} = |\{j \in \{1, \ldots, \ell\} : r_i^j = \beta\}|$, or
(2) $m_{i,\beta} = 1$ if $|\{j \in \{1, \ldots, \ell\} : r_i^j = \beta\}| \geq b$, where $b \in \{1, \ldots, \ell\}$. We will consider $b = \lfloor \ell/2 \rfloor + 1$ and $b = \lfloor \ell/2 \rfloor$ for our simulations in Sect. 4.

Example 1 Let $\mathbf{r} = (0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 2, 2) \in \mathbb{F}_3^{\ell n}$ a received word with $\ell = 5$ and $n = 3$, i.e. using matrix notation

$$\mathbf{r} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 2 & 2 \end{pmatrix}.$$

We consider the multiplicities assignment for this word, the non-zero multiplicities are:

$m_{1,0} = 5, m_{2,0} = 3, m_{2,1} = 1, m_{2,2} = 1, m_{3,0} = 2, m_{3,1} = 2, m_{3,2} = 1$ for multiplicity assignment (1).

$m_{1,0} = 1, m_{2,0} = 1$ for multiplicity assignment (2) with $b = 3$.

$m_{1,0} = 1, m_{2,0} = 1, m_{3,0} = 1, m_{3,1} = 1$ for multiplicity assignment (2) with $b = 2$.

For decoding, we will consider the soft-decision algorithm for C (for instance [8]) with these multiplicities. Trivially, we have that the computational complexity of this algorithm for C^ℓ equals the computational complexity of the soft-decoding algorithm for C with multiplicity assignments corresponding to the choice (1) or (2).

For instance, for the first multiplicity assignment it is $O(R^{\frac{1}{2}} n^2 m^5)$ using [8], where $R = k/n$ is the rate of a block and $m = max\{m_{j,\beta} : j = 1, \ldots, n \text{ and } \beta \in \mathbb{F}_q\}$. Thus, for the second multiplicity assignment, the complexity is $O(R^{\frac{1}{2}} n^2)$ using [8]. Notice that for decoding a repetition code with parameters $[n\ell, k, \ell d]$, we perform a list decoding of a Reed–Solomon code of length n, whilst a list decoding of a $[n\ell, k]$ Reed–Solomon code has complexity $O((k/\ell n)^{\frac{1}{2}} (\ell n)^2 m^5)$ using [9]. That is, our method is about $\ell^{3/2}$ times faster than decoding a Reed–Solomon code with the same parameters. One may consider other heuristics for multiplicity assignment between (1) and (2), that is, $0 \leq m_{i,\beta} \leq |\{j \in \{1,\dots,\ell\} : r_i^j = \beta\}|$. We have considered these two multiplicity assignments since they represent extremes between (1) maximizing the score and (2) minimizing the time complexity.

3.1 Multiplicity assignment (1)

For a word in C^ℓ, we consider its score using the first multiplicity assignment.

Lemma 1 *Let* $\mathbf{c} = (c,\dots,c)$ *be a sent codeword and let* $\mathbf{r} = \mathbf{c} + \mathbf{e}$ *be a received word with* $wt(\mathbf{e}) = t$. *Then*

$$score(\mathbf{c}) = \sum_{i=1}^{n} m_{i,c_i} = \ell n - t.$$

Proof Note that for $i = 1,\dots,n$, one has that $m_{i,c_i} = \ell - wt(e_i^1,\dots,e_i^l)$. Summing over the n positions gives the result. □

As the lemma shows, the score of the received word is a simple function of the number of errors t. In Sect. 6 we investigate the interplay between the number of conditions imposed, Eq. (3), and the score, we derive bounds for successful decoding of t errors, and we compare the bounds with the simulation results presented in Sect. 4.

3.2 Multiplicity assignment (2)

Using the second multiplicity assignment, the score of a received word might be lower than the one obtained using the first multiplicity assignment. Therefore, the error correction capability is smaller. However, the computational complexity is lower since $\max\{m_{j,\beta} : j = 1,\dots,n \text{ and } \beta \in \mathbb{F}_q\}$ is equal to 1. Note that if $m_{i,\beta} = 0$ for every $\beta \in \mathbb{F}_q$ then we are considering an erasure at position $i \in \{1\dots,n\}$.

We consider $b = \lfloor \ell/2 \rfloor + 1$ because if $b \geq \lfloor \ell/2 \rfloor + 1$, then, for $i \in \{1,\dots,n\}$, $m_{i,\beta} = 0$ for all $\beta \in \mathbb{F}_q$ but for, at most, one. Hence, we will have either an erasure (we interpolate with multiplicity zero at that position) or we interpolate with multiplicity one. With this choice for b the algorithm is very fast.

Remark 1 Let us compare our algorithm with some approaches in the bibliography. A decoding algorithm for a repeated code C^ℓ can be obtained decoding every block r_j of the received word r for $j = 1,\dots,\ell$ until the decoded block c'_j verifies that $(c'_j,\dots,c'_j)$ is at distance $\lfloor(\ell d - 1)/2\rfloor$ of the received word. This is the approach of [7] for quasi-cyclic codes, that is when C is cyclic. One could also consider C^ℓ as a convolutional code [3]. However, all these approaches will have bad performance due to the poor parameters of the repeated codes.

4 Computer experiments

We have compared the performance of our algorithm for the repetition code of a $[n, k]$ Reed–Solomon code over $\mathbb{F}_{p^v}$ to the list-decoding algorithm [9] for a $[n\ell, k]$ Reed–Solomon code over $\mathbb{F}_{p^u}$ where u is chosen so that $p^u > n\ell$. The information rate of the two codes is the same, although the $[n\ell, k]$ Reed–Solomon code uses larger symbol size. For each code we tested a range of values for t to determine the point where correction performance declined. We also compared the time required for decoding. In the repeated code case, we tested the algorithm described in the previous section for both multiplicity assignments. For the Reed–Solomon code over the larger field we used multiplicity 1 because for higher multiplicity, the algorithm is very slow.

We have implemented in MAGMA the decoding algorithm described in Sect. 3: given the codeword $c = (0, \dots, 0)$ in a $[n, k, d]$ Reed–Solomon code we consider the repeated codeword $(c, \dots, c) \in C^\ell$. In particular we choose $n = 63$, $\ell = 5$ and several values for k. We consider t errors in t uniformly distributed random positions among the ℓn positions, so it is possible that more than one error may occur in position i. We apply the algorithm in [8] with the prescribed multiplicities described using the multiplicity assignment (1) and (2) with $b = \lfloor \ell/2 \rfloor + 1 = 3$ and $b = \lfloor \ell/2 \rfloor = 2$. We ran the procedure 10,000 times each for different values of t. We declare success if the sent codeword is in the output list. For multiplicity assignment (2) the list has always size one since we are doing linear interpolation, and for multiplicity assignment (1) all the experiments produce size one as well.

1. Consider as block code the RS code with parameters $[63, 14, 50]$ over $\mathbb{F}_{2^6}$. The repeated code with $\ell = 5$ has parameters $[315, 14, 250]$. Our simulations in Tables 1 and 2 show that we can uniquely decode about 226 errors using multiplicity assignment (1), 183 errors using multiplicity assignment (2) with $b = 3$ and 218 errors using multiplicity assignment (2) with $b = 2$. However, the algo-

Table 1 List decoding $[315, 14, 250]$ repeated code over $\mathbb{F}_{2^6}$ with constituent $[63, 14, 50]$ RS code, multiplicity assignment (1)

t	224	225	226	227	228	229	230
$\frac{\text{Number of success}}{10{,}000}$	1	1	1	.9999	.9996	.9992	.9989
Time	1435.190	1424.100	1428.570	1324.950	1329.510	1322.360	1322.220

Table 2 List decoding $[315, 14, 250]$ repeated code over $\mathbb{F}_{2^6}$ with constituent $[63, 14, 50]$ RS code, multiplicity assignment (2)

t	182	183	184	185	217	218	219
$\frac{\text{Number of success}}{10{,}000}$, $b = 3$	1	1	9999	.9997	–	–	–
Time, $b = 3$	118.710	116.330	115.590	114.55	–	–	–
$\frac{\text{Number of success}}{10{,}000}$, $b = 2$	1	1	1	1	1	1	.9997
Time, $b = 2$	339.940	336.990	332.980	328.040	223.760	234.300	233.560

Table 3 List decoding [315, 14, 302] RS code over $\mathbb{F}_{2^9}$, multiplicity 1

t	229	230	231
$\frac{\text{Number of success}}{10{,}000}$	1	1	.3662
Time	250473	252184	258162

Table 4 List decoding [315, 40, 120] repeated code over $\mathbb{F}_{2^6}$ with constituent [63, 40, 24] RS code, multiplicity assignment (1)

t	153	154	155	156	157	158
$\frac{\text{Number of success}}{10{,}000}$	1	.9999	.9999	.9998	.9999	.9999
Time	1453.680	1454.820	1458.830	1455.610	1454.390	1450.540

Table 5 List decoding [315, 40, 120] repeated code over $\mathbb{F}_{2^6}$ with constituent [63, 40, 24] RS code, multiplicity assignment (2)

t	110	111	112	113	111	149	150	151
$\frac{\text{Number of success}}{10{,}000}$, $b = 3$	1	.9999	1	1	.9997	–	–	–
Time, $b = 3$	327.160	322.680	318.270	315.290	309.640	–	–	–
$\frac{\text{Number of success}}{10{,}000}$, $b = 2$	1	1	1	1	1	1	1	.9999
Time, $b = 2$	559.470	557.950	554.180	553.580	553.720	462.040	455.150	452.330

rithm using multiplicity assignment (2) with $b = 3$ is about 12 times (resp 6 times with $b = 2$) faster than the algorithm using multiplicity assignment (1).
We compare the previous algorithms with list decoding of the RS code over $\mathbb{F}_{2^9}$ with parameters [315, 14, 302]. Using multiplicity one we can decode about the same number of errors as with multiplicity assignment (1) but it is 176 times slower than the one with multiplicity assignment (1) and 1,077 times slower than the simulation with multiplicity assignment (2) and $b = 2$, see Table 3.

2. Consider as block code the RS code with parameters [63, 40, 24] over $\mathbb{F}_{2^6}$. The repeated code with $\ell = 5$ has parameters $[315, 40, 120]$. Our simulations in Tables 4 and 5 show that we can uniquely decode about 153 errors using multiplicity assignment (1), 110 errors using multiplicity assignment (2) with $b = 3$ (resp 150 errors with $b = 2$), but the algorithm using multiplicity assignment (2) is about 4.4 times faster with $b = 3$ (and 3.2 times faster with $b = 2$) than the algorithm using multiplicity assignment (1).
We compare the previous algorithms with list decoding of the RS code over $\mathbb{F}_{2^9}$ with parameters $[315, 40, 276]$. With multiplicity one we can decode more errors (177, see Table 6) than with multiplicity assignment (1) but it is 116 times slower than the one with multiplicity assignment (1) and it is 372 times slower than the one with multiplicity assignment (2) and $b = 2$.
3. Consider as block code the RS code with parameters [63, 54, 10] over $\mathbb{F}_{2^6}$. The repeated code, with $\ell = 5$, has parameters $[315, 54, 50]$. Our simulations in

Table 6 List decoding [315, 40, 276] RS code over $\mathbb{F}_{2^9}$, multiplicity 1

t	175	176	177	178
$\frac{\text{Number of success}}{10{,}000}$	1	1	1	.2912
Time	161480	169346	169805	151520

Table 7 List decoding [315, 54, 50] repeated code over $\mathbb{F}_{2^6}$ with constituent [63, 54, 10] RS code, multiplicity assignment (1)

t	93	94	95	96	97	98	99
$\frac{\text{Number of success}}{10{,}000}$	1	1	.9999	.9999	1	.9999	.9997
Time	1379.150	1372.370	1381.790	1388.470	1398.770	1405.480	1418.600

Table 8 List decoding [315, 54, 50] repeated code over $\mathbb{F}_{2^6}$ with constituent [63, 54, 10] RS code, multiplicity assignment (2)

t	60	61	62	64	88	89	90
$\frac{\text{Number of success}}{10{,}000}$, $b = 3$	1	1	1	.9997	–	–	–
Time, $b = 3$	539.10	540.940	530.780	527.040	–	–	–
$\frac{\text{Number of success}}{10{,}000}$, $b = 2$	1	1	1	1	1	1	.9999
Time, $b = 2$	614.080	618.220	613.190	611.620	607.170	605.460	596.280

Table 9 List decoding [315, 54, 262] RS code over $\mathbb{F}_{2^9}$, multiplicity 1

t	150	155	156	157
$\frac{\text{Number of success}}{10{,}000}$	1	1	1	.2657
Time	115934	124500	125797	129186

Tables 7 and 8 show that we can uniquely decode about 94 errors using multiplicity assignment (1) and 62 errors using multiplicity assignment (2) with $b = 3$ (resp 89 with $b = 2$). However, the algorithm using multiplicity assignment (2) with $b = 3$ is about 2.5 times faster (2.2 times faster with $b = 2$) than the algorithm using multiplicity assignment (1).
We compare the previous algorithms with list decoding of the RS code over $\mathbb{F}_{2^9}$ with parameters [315, 54, 262]. Using multiplicity one, we can decode more errors, about 156, than with multiplicity assignment (1) but it is 89 times slower than the one with multiplicity assignment (1) and it is 207 times slower than with multiplicity assignment method (2) and $b = 2$, see Table 9.

4. Finally, we consider an example over a field with characteristic 3. Consider as block code the RS code with parameters [26, 14, 13] over $\mathbb{F}_{3^3}$. The repeated code with $\ell = 5$ has parameters [130, 14, 65]. Our simulations in Tables 10 and 11 show that we can uniquely decode 65 errors using multiplicity assignment (1) and 46 errors using multiplicity assignment (2) with $b = 3$ (resp 53 with $b = 2$).

Table 10 List decoding [130, 14, 65] repeated code over $\mathbb{F}_{3^3}$ with constituent [26, 14, 13] RS code, multiplicity assignment (1)

t	62	63	64	65	66
$\frac{\text{Number of success}}{10{,}000}$	1	1	1	1	.9998
Time	254.390	254.590	255.620	254.980	256.400

Table 11 List decoding [130, 14, 65] repeated code over $\mathbb{F}_{3^3}$ with constituent [26, 14, 13] RS code, multiplicity assignment (2)

t	46	47	48	49	52	53	54
$\frac{\text{Number of success}}{10{,}000}$, $b = 3$	1	.9999	1	.9999	–	–	–
Time, $b = 3$	55.360	49.280	51.850	45.150	–	–	–
$\frac{\text{Number of success}}{10{,}000}$, $b = 2$	–	–	–	–	1	1	.9999
Time, $b = 2$	–	–	–	–	97.230	96.940	96.540

Table 12 List decoding [130, 14, 117] RS code over $\mathbb{F}_{3^5}$, multiplicity 1

t	75	76	77	78
$\frac{\text{Number of success}}{10{,}000}$	1	1	1	0.2769
Time	10055	10244	10320	10193

However, the algorithm using multiplicity assignment (2) with $b = 3$ is about 4.7 times faster (2, 7 times faster with $b = 2$) than the algorithm using multiplicity assignment (1).

We compare the previous algorithms with list decoding of the RS code over $\mathbb{F}_{3^5}$ with parameters [130, 14, 117]. Using multiplicity one, we can decode more errors (77 errors, see table 12) than with multiplicity assignment (1) but it is 40 times slower than the one with multiplicity assignment (1) and it is 106 times slower than with multiplicity assignment (2) and $b = 2$, see Table 12.

We consider a list decoding algorithm with multiplicity 1 for the RS codes due to the fact that the times obtained in Tables 3, 6, 9, 12 show that higher multiplicity would be impracticable. These experiments clearly show advantages to using a repeated code and our method for decoding as compared to using a low rate Reed–Solomon code. The decoding complexity is much lower and the correction capability is quite similar—even better in the very low rate example—for the repeated code. Although the repeated code does not have good parameters, it is because a rare few codewords are "close" to the sent codeword. However, these have little affect on the decoding performance.

5 Bounds for the correction capability using multiplicity assignment (2)

We consider the decoding capability of this algorithm for the repetition code of a Reed–Solomon code with $b = \lfloor \ell/2 \rfloor + 1$ and $b = \lfloor \ell/2 \rfloor$. The decoding capability of the soft-decoding algorithm for Reed–Solomon codes can be found in [8, Section IV]. However, the bounds are obtained in terms of the interpolation multiplicities and we cannot infer a bound in terms of the weight of the error vector. In order to obtain such a bound, we should assume that the error vector verifies an additional condition for performing our analysis.

Theorem 1 *The algorithm introduced in* Sect. 3 *using the second multiplicity assignment for a Reed–Solomon repetition code* $[n\ell, k, \ell(n-k+1)]$ *can decode at least the following number of errors if the error vector* $\mathbf{e}$ *verifies the following assumption: for every* $i \in \{1, \ldots, n\}$ *and* $\beta \in \mathbb{F}_q \setminus \{0\}$,

$$\#\{e_i^j = \beta : j = 1, \ldots, \ell\} \le b - 1.$$

– *For* $b = \lfloor \ell/2 \rfloor + 1$ *and* ℓ *odd,*

$$(n-k)\left(\left\lfloor \frac{\ell}{2} \right\rfloor + 1\right) + \left\lfloor \frac{\ell}{2} \right\rfloor \text{ errors.}$$

– *For* $b = \lfloor \ell/2 \rfloor + 1$ *and* ℓ *even,*

$$(n-k)\left\lfloor \frac{\ell}{2} \right\rfloor + \left\lfloor \frac{\ell}{2} \right\rfloor - 1 \text{ errors.}$$

– *For* $b = \lfloor \ell/2 \rfloor$ *and* ℓ *odd,*

$$(n-k)\left(\left\lfloor \frac{\ell}{2} \right\rfloor + 2\right) + \left(\left\lfloor \frac{\ell}{2} \right\rfloor + 1\right) \text{ errors.}$$

– *For* $b = \lfloor \ell/2 \rfloor$ *and* ℓ *even,*

$$(n-k)\left(\left\lfloor \frac{\ell}{2} \right\rfloor + 1\right) + \left\lfloor \frac{\ell}{2} \right\rfloor \text{ errors.}$$

Proof Notice that the constituent code is a MDS code. Under the assumption that $\#\{e_i^j = \beta : j = 1, \ldots, \ell\} \le b - 1$ for every $i \in \{1, \ldots, n\}$ and $\beta \in \mathbb{F}_q \setminus \{0\}$ we guarantee that we only have erasures (we have no errors) and we assign to them multiplicity zero. An erasure MDS code can correctly decode a received word if k symbols are non-corrupted, in our setting, this means that we have assigned multiplicity one. The point (α^i, r_i^j) is assigned multiplicity one if and only if b blocks have the same value in the ith position. Therefore $(n-k)$ positions can be corrupted in $\lfloor \ell/2 \rfloor + 1$ blocks and we still can decode it correctly if $b = \lfloor \ell/2 \rfloor + 1$ and ℓ is odd, $\lfloor \ell/2 \rfloor$ if $b = \lfloor \ell/2 \rfloor + 1$ and ℓ is even, $(\lfloor \ell/2 \rfloor + 2)$ if $b = \lfloor \ell/2 \rfloor$ and ℓ is odd, $(\lfloor \ell/2 \rfloor + 1)$ if $b = \lfloor \ell/2 \rfloor$ and ℓ is even.

In the other k positions there could be at most $\lfloor \ell/2 \rfloor$ errors to guarantee success, i.e. to obtain an erasure in such position, if $b = \lfloor \ell/2 \rfloor + 1$ and ℓ is odd, $\lfloor \ell/2 \rfloor - 1$ if

$b = \lfloor \ell/2 \rfloor + 1$ and ℓ is even, $\lfloor \ell/2 \rfloor + 1$ if $b = \lfloor \ell/2 \rfloor$ and ℓ is odd, $\lfloor \ell/2 \rfloor$ if $b = \lfloor \ell/2 \rfloor$ and ℓ is even.

Summing up all these values we get the above bound. □

Remark 2 Let us compute these bounds for the examples in previous section.

- For the [63, 14, 50] RS code and $\ell = 5$ and $b = 3$ Theorem 1 tell us that we can decode 149 errors while if $b = 2$ we can decode 199 errors. If we check the computer experiments in Table 2 we can successfully decode 183 errors for $b = 3$ and 218 errors for $b = 2$ respectively.
- For the [63, 40, 50] RS code and $\ell = 5$ and $b = 3$ Theorem 1 tell us that we can decode 71 errors while if $b = 2$ we can decode 95 errors. If we check the computer experiments in Table 5 we can successfully decode 113 errors for $b = 3$ and 150 errors for $b = 2$ respectively.
- For the [63, 54, 50] RS code and $\ell = 5$ and $b = 3$ Theorem 1 tell us that we can decode 29 errors while if $b = 2$ we can decode 39 errors. If we check the computer experiments in Table 8 we can successfully decode 61 errors for $b = 3$ and 89 errors for $b = 2$ respectively.
- For the [26, 14, 13] RS code and $\ell = 5$ and $b = 3$ Theorem 1 tell us that we can decode 38 errors while if $b = 2$ we can decode 51 errors. If we check the computer experiments in Table 11 we can successfully decode 53 errors for $b = 2$ and 46 errors for $b = 3$ respectively.

Therefore, we claim that we can use our bounds as a conservative estimate of the real decoding capacity, especially for low rate codes, independently of the assumptions.

Note that our assumption in Theorem 1 for the error vector $\mathbf{e}$ will hold with a high probability if the field is not too small: the higher the base field is, the more unlikely is that two error positions e_i^j and $e_i^{j'}$ are equal, for $i = 1, \ldots, n$. We compute now when this algorithm can decode more errors than a $[n\ell, k, n\ell - k + 1]$ Reed–Solomon code with a unique decoding algorithm. Notice that we perform list decoding of a Reed–Solomon code over $\mathbb{F}_q$ and we compare it with a unique decoding of a $[n\ell, k, n\ell - k + 1]$ MDS code over $\mathbb{F}_q$ (if it exists) or over a higher field.

- If $b = \lfloor \ell/2 \rfloor + 1$ and ℓ is odd, i.e., $\ell = 2\ell' + 1$ where $\ell' = \lfloor \ell/2 \rfloor$.

$$(n-k)\left(\left\lfloor \frac{\ell}{2} \right\rfloor + 1\right) + \left\lfloor \frac{\ell}{2} \right\rfloor \geq \left\lfloor \frac{n\ell - k + 1}{2} \right\rfloor \geq \frac{n\ell - k + 1}{2}$$

We compare the leftmost and rightmost sides of the formula. We have that

$$\begin{aligned} &2(n-k)(\ell'+1) + 2\ell' \geq 2n\ell' + n - k + 1 \\ &2n\ell' + 2n - 2k(\ell'+1) + 2\ell' \geq 2n\ell' + n - k + 1 \\ &n + 2\ell' - 1 \geq k(2\ell'+1) \\ &k \leq \frac{n + 2\ell' - 1}{\ell} \end{aligned}$$

For example if $n = 63$, $\ell = 5$, $\ell' = 2$ then for $k \leq 13$ we can correctly decode at least up to the half of the minimum distance of the corresponding Reed–Solomon code. According to our computations for $k = 14$ we can indeed decode about 187 errors but a Reed–Solomon code can correct 150 errors.

Analogously, we have that:

– If $b = \lfloor \ell/2 \rfloor + 1$ and ℓ is even, i.e., $\ell = 2\ell'$ where $\ell' = \lfloor \ell/2 \rfloor$, then

$$k \leq \frac{2\ell' - 3}{2\ell' - 1} < 1.$$

Therefore, we can conclude that one should not consider ℓ even for $b \geq \lfloor \ell/2 \rfloor + 1$.

– If $b = \lfloor \ell/2 \rfloor$ and ℓ is odd, i.e., $\ell = 2\ell' + 1$ where $\ell' = \lfloor \ell/2 \rfloor$, then

$$k \leq \frac{3n + 2\ell' + 1}{2\ell' + 3}$$

For example if $n = 63$, $\ell = 5$, $\ell' = 2$ then for $k \leq 27$ we can correctly decode at least up to the half of the minimum distance of the corresponding Reed–Solomon code. According to our computations for $k = 14$ we can indeed decode about 219 errors but a Reed–Solomon code can correct 150 errors.

– If $b = \lfloor \ell/2 \rfloor$ and ℓ is even, i.e., $\ell = 2\ell'$ where $\ell' = \lfloor \ell/2 \rfloor$, then

$$k \leq \frac{2n + 2\ell' - 1}{2\ell' + 1}.$$

For example if $n = 63$, $\ell = 4$, $\ell' = 2$ then for $k \leq 25$ we can correctly decode at least up to the half of the minimum distance of the corresponding Reed–Solomon code.

6 Bounds for the correction capability using multiplicity assignment (1)

As mentioned in Sect. 2, the interpolation problem consists in finding a bivariate polynomial, $Q(x, y)$, passing through the points $p_{i,\beta}$ with multiplicity $m_{i,\beta}$, where the multiplicities are described in Sect. 3. Therefore, we compute a Gröbner basis of the ideal I_M, defined in (1), with respect to the $(1, k-1)$-weighted degree and consider the smallest element in this basis. Succesful decoding is ensured when for some integer d the following two conditions are satisfied:

(i) The number of monomials of $(1, k-1)$ degree at most d is larger than the number of conditions imposed.
(ii) The score of the sent codeword c is larger than d.

The first item ensures the existence of a polynomial $f \in I_M$ of weighted degree at most d and the second ensures that $y - h_c$ is a factor of f.

In the rest of this section we analyze when both conditions are simultaneously satisfied in the context of quasi-cyclic codes with multiplicity assignment (2). Let **r** be the received word, let t be the number of errors, and for each $i = 1, \ldots, n$ let τ_i be the number of errors in position i. Thus, $\sum_{i=1}^{n} \tau_i = t$.

Let us start with condition (ii). By Lemma 1 the score of $\mathbf{r}$ is $\ell n - t$. Let a, b be the unique integers satisfying $0 \leq b < k-1$ and

$$\ell n - t - 1 = a(k-1) + b$$

Now consider condition (i). For $a \geq 0$ and $0 \leq b \leq k-2$ let $P_{a,b}$ be the set of monomials with $(1, k-1)$-weighted degree lower than or equal to $a(k-1)+b$. The number of monomials in $P_{a,b}$ is

$$(k-1) + 2(k-1) + \cdots + a(k-1) + (a+1)(b+1) = \frac{a(a+1)}{2}(k-1) + (a+1)(b+1). \tag{4}$$

Thus, comparing (4) with (3) we get the following condition:

$$(k-1)\frac{a(a+1)}{2} + (b+1)(a+1) \geq \sum_{i=1}^{n} \sum_{\beta \in \mathbb{F}_q} \binom{m_{i,\beta}+1}{2} + 1 \tag{5}$$

where M is the multiplicity matrix derived from $\mathbf{r}$ according to multiplicity assignment (2). In order to analyze the multiplicity matrix we will make a simplifying assumption: If two errors occur at position i and blocks j, k then $r_i^j \neq r_i^k$. That is,

$$e_i^j \neq e_i^k. \tag{6}$$

If the base field is large and ℓ is small, this assumption is realistic.

Under the above assumption, for position i there are $\ell - \tau_i$ blocks that have the correct value and τ_i that are incorrect, but unequal to one another. The number of conditions imposed by the correct blocks is is $\binom{\ell-\tau_i}{2}$, while the incorrect blocks impose one condition each, for a total of τ_i. Thus the total number of conditions imposed is

$$\begin{aligned}\sum_{i=1}^{n}\left(\binom{\ell-\tau_i}{2} + \tau_i\right) &= \sum_{i=1}^{n}\left(\frac{(\ell-\tau_i)(\ell-\tau_i+1)}{2} + \tau_i\right)\\ &= \sum_{i=1}^{n}\left(\frac{(\ell)(\ell+1)}{2} - \tau_i(\ell-1) + \frac{\tau_i(\tau_i-1)}{2}\right)\\ &= n\frac{(\ell)(\ell+1)}{2} - t(\ell-1) + \sum_{i=1}^{n}\frac{\tau_i(\tau_i-1)}{2}\end{aligned}$$

Notice that the final sum $A = \sum_{i=1}^{n} \dfrac{\tau_i(\tau_i-1)}{2}$ is the only one that depends on the distribution of the errors. We now consider three cases. The term A is minimized when the τ_i are distributed as evenly possible. Let $t = nq_1 + s_1$ with $0 \leq s_1 < n$ and assume that s_1 positions have $q_1 + 1$ errors while $n - s_1$ positions have q_1 errors. The final term is then

$$A_{\min} = s_1 \frac{(q_1+1)q_1}{2} + (n-s_1)\frac{(q_1-1)q_1}{2} = n\frac{q_1(q_1-1)}{2} + s_1 q_1$$

The final term A is maximized when the errors are consolidated into as few positions as possible. Let $t = \ell q_2 + s_2$ with $0 \le s_2 < \ell$ and assume that q_2 positions have ℓ errors and that one position has s_2 errors. The final term is then

$$A_{\max} = q_2 \frac{\ell(\ell-1)}{2} + \frac{s_2(s_2-1)}{2}$$

Finally we consider the expected value of the final term A, subject to the t error positions being randomly chosen from $\{(i, j) : i \in \{1, \dots, n\}, j \in \{1, \dots, \ell\}\}$. The probability of any particular $\tau_1, \dots, \tau_n$ occurring is

$$\frac{\prod_{i=1}^{n} \binom{\ell}{\tau_i}}{\binom{n\ell}{t}}.$$

Let $\mathbf{x}$ stand for indeterminates $x_1, \dots, x_n$ and let $\mathbf{1}$ be an n-tuple with 1 in each entry. Let $|\tau| = \sum_{i=1}^{n} \tau_i$. Consider the generating function

$$B(\mathbf{x}, s) = \prod_{i=1}^{n} (1 + s x_i)^{\ell} \tag{7}$$

The term in s^t in $B(\mathbf{x}, s)$ is

$$\sum_{|\tau|=t} \prod_{i=1}^{n} \binom{\ell}{\tau_i} (s x_i)^{\tau_i} = s^t \sum_{|\tau|=t} \prod_{i=1}^{n} \binom{\ell}{\tau_i} (x_i)^{\tau_i}$$

Taking the second derivatives of $B(\mathbf{x}, s)$ with respect to the x_i one finds that the term in s^t in $\sum_{i=1}^{n} \frac{\partial^2 B}{\partial x_i^2}(\mathbf{x}, s)$ is

$$s^t \sum_{|\tau|=t} \left(\prod_{i=1}^{n} \binom{\ell}{\tau_i} \right) \sum_{i=1}^{n} \tau_i(\tau_i - 1)(x_i)^{\tau_i - 2}$$

Evaluating at $\mathbf{x} = \mathbf{1}$ we get the term in s^t in $\sum_{i=1}^{n} \frac{\partial^2 B}{\partial x_i^2}(\mathbf{1}, s)$ is

$$s^t \sum_{|\tau|=t} \left(\prod_{i=1}^{n} \binom{\ell}{\tau_i} \right) \sum_{i=1}^{n} \tau_i(\tau_i - 1)$$

Table 13 Bounds for the correction capability using multiplicity assignment (1)

Code	Max	Exp	Min	Range
$[63, 14]_{\mathbb{F}_{2^6}}$	203	223	227	226–230
$[63, 40]_{\mathbb{F}_{2^6}}$	89	130	142	153–158
$[63, 54]_{\mathbb{F}_{2^6}}$	38	69	81	94–99
$[26, 14]_{\mathbb{F}_{3^3}}$	49	64	69	65–66

Thus the expected value of the final term A is the coefficient of s^t in $(2\binom{n\ell}{t})^{-1}\sum_{i=1}^{n} \frac{\partial^2 B}{\partial x_i^2}(\mathbf{1}, s)$

Computing from (7) we have

$$\sum_{i=1}^{n} \frac{\partial^2 B}{\partial x_i^2}(\mathbf{x}, s) = s^2 \sum_{i=1}^{n} \ell(\ell-1)(1+sx_i)^{\ell-2} \prod_{\substack{j=1\\ j\neq i}}^{n} (1+sx_j)^{\ell}$$

$$\begin{aligned}\sum_{i=1}^{n} \frac{\partial^2 B}{\partial x_i^2}(\mathbf{1}, s) &= s^2 \sum_{i=1}^{n} \ell(\ell-1)(1+s)^{n\ell-2}\\ &= s^2 n\ell(\ell-1)(1+s)^{n\ell-2}\end{aligned}$$

The coefficient of s^t in $(2\binom{n\ell}{t})^{-1}\sum_{i=1}^{n} \frac{\partial^2 B}{\partial x_i^2}(\mathbf{1}, s)$ is therefore

$$A_{\text{exp}} = n\ell(\ell-1)\frac{(n\ell-2)!}{(t-2)!(n\ell-t)!}\frac{(n\ell-t)!t!}{2(n\ell)!} = \frac{t(t-1)(\ell-1)}{2(n\ell-1)}$$

In the following table for each code considered in Sect. 4 we show the maximum value t for which the algorithm is able to decode in three cases: for the worst scenario, for the expected scenario, and for the best scenario (i.e. using A_{max}, A_{exp}, A_{min}). We also list the range where experiments showed that decoding capability began to decline. It is interesting to see that the experimental results are somewhat better, even better than the best scenario (using A_{min}), particularly at higher rate. In the simulations, the number of monomials less than the leading term of the minimal $Q(x, y)$ that was computed is noticeably smaller than the number of conditions imposed. In other words, the conditions imposed by interpolation are not independent. We have no explanation for this phenomenon, but it appears to be the key to the performance beyond our estimates (Table 13).

7 Conclusion

An efficient list-decoding algorithm for repeated codes, in particular for repeated Reed–Solomon codes is presented. The theoretical and experimental results show that decoding repeated codes with this algorithm yields surprisingly good error correction performance, nearly comparable to that of a Reed–Solomon codes over a larger field. Furthermore, the computational burden of the repeated code is much lower because of the smaller field size.

References

1. Alekhnovich, M.: Linear Diophantine equations over polynomials and soft decoding of Reed–Solomon codes. IEEE Trans. Inf. Theory **51**(7), 2257–2265 (2005)
2. Bosma, W., Cannon, J., Playoust, C.: The magma algebra system. I. The user language. J. Symb. Comput. **24**(3–4), 235–265 (1997)
3. Dumer I.I.: Concatenated codes and their multilevel generalizations. In: Pless, V. Huffman W. C. (eds.) Handbook of Coding Theory, vol. I, II, pp. 1911–1988. North-Holland, Amsterdam (1998)
4. Elias, P.: List decoding for noisy channels. Research Laboratory of Electronics, Massachusetts Institute of Technology, Cambridge, Mass., Rep. No. 335 (1957)
5. Guruswami, V., Sudan, M.: Improved decoding of Reed–Solomon and algebraic-geometry codes. IEEE Trans. Inf. Theory **45**(6), 1757–1767 (1999)
6. Koetter, R., Vardy, A.: Algebraic soft-decision decoding of Reed–Solomon codes. IEEE Trans. Inf. Theory **49**(11), 2809–2825 (2003)
7. Lally, K.: Quasicyclic codes—some practical issues. In: Proceedings. 2002 IEEE International Symposium on Information Theory (2002)
8. Lee, K., O'Sullivan, M.E.: An interpolation algorithm using Groebner bases for soft-decision decoding of Reed–Solomon codes. In: Proceedings. 2006 IEEE International Symposium on Information Theory, vol. 7, pp. 2032–2036 (2006)
9. Lee, K., O'Sullivan, M.E.: List decoding of Reed–Solomon codes from a Gröbner basis perspective. J. Symb. Comput. **43**(9), 645–658 (2008)
10. MacWilliams, F.J., Sloane, N.J.A.: The theory of error-correcting codes, vol. 16. North-Holland Publishing Co., North-Holland Mathematical Library, Amsterdam (1977)
11. Sudan, M.: Decoding of Reed–Solomon codes beyond the error-correction bound. J. Complexity. **13**(1), 180–193 (1997)
12. Wozencraft, J.M.: List decoding. In: Quarterly Progress Report, pp. 90–95. Research Laboratory of Electronics, MIT, Cambridge, MA (1958)