List Decoding of Matrix-Product Codes from nested codes: an application to Quasi-Cyclic codes*

Fernando Hernando[1]

Department of Mathematics, Universidad Jaume I,
Campus Riu Sec, 12071, Castellón de la plana, Spain
carrillf@mat.uji.es

Tom Høholdt

DTU-Mathematics, Technical University of Denmark,
Matematiktorvet, Building 303, 2800 Kgs. Lyngby, Denmark
T.Hoeholdt@mat.dtu.dk

Diego Ruano

Department of Mathematical Sciences, Aalborg University,
Fr. Bajersvej 7G, 9920-Aalborg Øst, Denmark
diego@math.aau.dk

Abstract

A list decoding algorithm for matrix-product codes is provided when $C_1, \ldots, C_s$ are nested linear codes and A is a non-singular by columns matrix. We estimate the probability of getting more than one codeword as output when the constituent codes are Reed-Solomon codes. We extend this list decoding algorithm for matrix-product codes with polynomial units, which are quasi-cyclic codes. Furthermore, it allows us to consider unique decoding for matrix-product codes with polynomial units.

1 Introduction

Matrix-product codes, $[C_1 \cdots C_s] \cdot A$, are a generalization of several classic codes constructions of codes from old ones [2, 15]. For instance, they extend the $(u|u+v)$-construction. An algorithm for unique decoding when the codes are nested, $C_1 \supset \cdots \supset C_s$, and A has a certain property, called non-singular by columns, was provided in [8]. The algorithm decodes up to half of the minimum

*This work was supported in part 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 Claude Shannon Institute, Science Foundation Ireland Grant 06/MI/006 and by Spanish MEC Grant MTM2007-64704. *2000 Mathematics Subject Classification.* Primary: 94B05; Secondary: 94B35. *Keywords:* Linear Code, Matrix-Product Code, List Decoding, Quasi-Cyclic Code.

[1]INSPIRE fellow funding received from the Irish Research Council for Science, Engineering and Technology.

distance, assuming that we have a decoding algorithm for C_i that decodes up to half of its minimum distance, for every i.

List decoding was introduced by Elias [5] and Wozencraft [16]. The list decoder is a relaxation over unique decoding that allows the decoder to produce a list of codewords as answers. It can uniquely decode beyond half of the minimum distance in some cases or to produces a list of codewords.

In 1997, Sudan presented a polynomial time algorithm for decoding low rate Reed-Solomon codes beyond the classical $\frac{d}{2}$ bound. Later [7], Guruswami and Sudan provided a significantly improved version of list decoder which can correct codes of any rates. Recently, Lee and O'Sullivan provide a list decoding algorithm based on the computation of a Gröbner basis of a module [13] and Beelen and Brander provide an algorithm that has linear complexity in the code length [1].

In this paper we consider a list decoding algorithm for matrix-product codes which is an extension of the algorithm in [8]. The algorithm in [8] assumes a known decoding algorithm for every constituent code C_i that decodes up to half of its minimum distance, for this algorithm, we assume that the decoding algorithm is a list-decoding algorithm. Moreover, it is also required that $C_1, \ldots, C_s$ are nested and A is non-singular by columns. The extension is natural, but, we believe, it is non-trivial task since we had to modify the algorithm to deal with lists of codewords, compute the error bound τ and prove the correctness of the algorithm, among others.

Matrix-Product codes are generalized concatenated codes [2, 4], which have an efficient decoding algorithm [6]. However, this algorithm cannot be successfully applied if the matrix A is small, as it is in practice for Matrix-product codes (see Remark 3.4).

The probability of getting more than one codeword as output of a list decoding algorithm for Reed-Solomon codes was bounded in [14]. In section 4, we use this computation to estimate an upper bound of the probability of getting more than one codeword as output, when $C_1, \ldots, C_s$ are Reed-Solomon codes. The algorithm in section 3 can become computationally intense, an optimal situation arises considering $s = l = 2$.

In section 5 we extend the algorithm in section 3 for matrix-product codes with polynomial units [9], which are quasi-cyclic codes. Quasi-cyclic codes became important after it was shown that some codes in this class meet a modified Gilbert-Varshamov bound [10], however there are no general fast algorithms for decoding them. In [9], many of these codes with good parameters were obtained. Using list decoding of matrix-product codes with polynomial units we can uniquely decode these codes up to the half of the minimum distance.

2 Matrix-Product Codes

A matrix-product code is a construction of a code from old ones.

Definition 2.1. Let $C_1, \ldots, C_s \subset \mathbb{F}_q^m$ be linear codes of length m and a matrix $A = (a_{i,j}) \in \mathcal{M}(\mathbb{F}_q, s \times l)$, with $s \leq l$. The **matrix-product code** $C = [C_1 \cdots C_s] \cdot A$ is the set of all matrix-products $[c_1 \cdots c_s] \cdot A$ where $c_i \in C_i$ is an $m \times 1$ column vector $c_i = (c_{1,i}, \ldots, c_{m,i})^T$ for $i = 1, \ldots, s$. Therefore, a typical codeword $\mathbf{c}$ is

$$\mathbf{c} = \begin{pmatrix} c_{1,1}a_{1,1} + \cdots + c_{1,s}a_{s,1} & \cdots & c_{1,1}a_{1,l} + \cdots + c_{1,s}a_{s,l} \\ \vdots & \ddots & \vdots \\ c_{m,1}a_{1,1} + \cdots + c_{m,s}a_{s,1} & \cdots & c_{m,1}a_{1,l} + \cdots + c_{m,s}a_{s,l} \end{pmatrix}. \quad (1)$$

Clearly the i-th column of any codeword is an element of the form $\sum_{j=1}^{s} a_{j,i}c_j \in \mathbb{F}_q^m$, therefore reading the entries of the $m \times l$-matrix above in column-major order, the codewords can be viewed as vectors of length ml,

$$\mathbf{c} = \left(\sum_{j=1}^{s} a_{j,1}c_j, \ldots, \sum_{j=1}^{s} a_{j,l}c_j \right) \in \mathbb{F}_q^{ml}. \quad (2)$$

From the above construction it follows that a generator matrix of C is of the form:

$$G = \begin{pmatrix} a_{1,1}G_1 & a_{1,2}G_1 & \cdots & a_{1,s}G_1 & \cdots & a_{1,l}G_1 \\ a_{2,1}G_2 & a_{2,2}G_2 & \cdots & a_{2,s}G_2 & \cdots & a_{2,l}G_2 \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ a_{s,1}G_s & a_{s,2}G_s & \cdots & a_{s,s}G_s & \cdots & a_{s,l}G_s \end{pmatrix},$$

where G_i is a generator matrix of C_i, $i = 1, \ldots, s$. Moreover, if C_i is a $[m, k_i, d_i]$ code then one has that $[C_1 \cdots C_s] \cdot A$ is a linear code over $\mathbb{F}_q$ with length lm and dimension $k = k_1 + \cdots + k_s$ if the matrix A has full rank and $k < k_1 + \cdots + k_s$ otherwise.

Let us denote by $R_i = (a_{i,1}, \ldots, a_{i,l})$ the element of $\mathbb{F}_q^l$ consisting of the i-th row of A, for $i = 1, \ldots, s$. We denote by D_i the minimum distance of the code C_{R_i} generated by $\langle R_1, \ldots, R_i \rangle$ in $\mathbb{F}_q^l$. In [15] the following lower bound for the minimum distance of the matrix-product code C is obtained,

$$d(C) \geq \min\{d_1D_1, d_2D_2, \ldots, d_sD_s\}, \quad (3)$$

where d_i is the minimum distance of C_i. If $C_1, \ldots, C_s$ are nested codes, $C_1 \supset \cdots \supset C_s$, the previous bound is sharp [8].

In [2], the following condition for the matrix A is introduced.

Definition 2.2. [2] Let A be a $s \times l$ matrix and A_t be the matrix consisting of the first t rows of A. For $1 \leq j_1 < \cdots < j_t \leq l$, we denote by $A(j_1, \ldots, j_t)$ the $t \times t$ matrix consisting of the columns $j_1, \ldots, j_t$ of A_t.

A matrix A is non-singular by columns if $A(j_1, \ldots, j_t)$ is non-singular for each $1 \leq t \leq s$ and $1 \leq j_1 < \cdots < j_t \leq l$. In particular, a non-singular by columns matrix A has full rank.

Moreover, if A is non-singular by columns and $C_1 \supset \cdots \supset C_s$, we have $d(C) = \min\{ld_1, (l-1)d_2, \ldots, (l-s+1)d_s\}$ [8].

In [8] were presented a decoding algorithm for the matrix-product code $C = [C_1 \cdots C_s] \cdot A \subset \mathbb{F}_q^{ml}$, with A non-singular by columns and $C_1 \supset \cdots \supset C_s$, assuming that we have a decoding algorithm for C_i, for $i = 1, \ldots, s$. The algorithm in [8] decodes up to half of the minimum distance. In next section we provide a list decoding algorithm for such codes, assuming that we have a list decoding algorithm for C_i, $i = 1, \ldots, s$.

3 List Decoding Algorithm for matrix-product codes

Let $C \subset \mathbb{F}_q^n$ and $\tau > 1$. For $r \in \mathbb{F}_q^n$, a list decoding algorithm with error bound τ provides a list with all codewords in C that differ from r in at most τ places. If $\tau \leq \lfloor \frac{d-1}{2} \rfloor$, it will result into unique decoding.

We present a list decoding algorithm for a class of matrix-product codes, it is an extension of [8, Algorithm 1]. Namely, we consider s nested linear codes $C_1, \ldots, C_s \subset \mathbb{F}_q^m$ and a non-singular by columns matrix $A \in \mathcal{M}(\mathbb{F}_q, s \times l)$, where $s \leq l$. We provide a list decoding algorithm for the matrix-product code $C = [C_1 \cdots C_s] \cdot A \subset \mathbb{F}_q^{ml}$, assuming that we have a list decoding algorithm LDC_i for C_i with error bound τ_i. In particular, each LDC_i answers an empty list if there is no codeword in C_i within distance τ_i of the received word.

Our list algorithm for C decodes up to

$$\tau = \min\{l\tau_1 + (l-1), (l-1)\tau_2 + (l-2), \ldots, (l-s+1)\tau_s + l - s\}. \quad (4)$$

We first describe the main steps in our decoding algorithm. The algorithm is outlined as a whole in procedural form in Algorithm 1.

Consider the codeword $\mathbf{c} = (\sum_{j=1}^s a_{j,1}c_j, \ldots, \sum_{j=1}^s a_{j,l}c_j)$, where $c_j \in C_j$, for all j. Suppose that $\mathbf{c}$ is sent and that we receive $\mathbf{p} = \mathbf{c} + \mathbf{e}$, where $\mathbf{e} = (e_1, e_2, \ldots, e_l) \in \mathbb{F}_q^{ml}$ is an error vector. We denote by $p_i = \sum_{j=1}^s a_{j,i}c_j + e_i \in \mathbb{F}_q^m$ the i-th block of $\mathbf{p}$, for $i = 1, \ldots, l$. Let $\{i_1, \ldots, i_s\} \subset \{1, \ldots, l\}$ be an ordered subset of indices. We now also suppose that $\mathbf{e}$ satisfies the extra property that

$$wt(e_{i_j}) \leq \tau_j \text{ for all } j \in \{1, \ldots, s\}. \quad (5)$$

Since $C_1 \supset \cdots \supset C_s$, each block $\sum_{j=1}^s a_{j,i}c_j$ of $\mathbf{c}$ is a codeword of C_1. Therefore, we decode the i_1-th block p_{i_1} of $\mathbf{p}$ using LDC_1 and we obtain a list L_1. Since $wt(e_{i_1}) \leq \tau_1$, we have $\sum_{j=1}^s a_{j,i_1}c_j \in L_1$. In practice we do not know which one of the elements in L_1 is $\sum_{j=1}^s a_{j,1}c_j$, therefore we should consider the following computations for every element in L_1. Assume now that we consider $\sum_{j=1}^s a_{j,i_1}c_j \in L_1$, hence we obtain $e_{i_1} = p_{i_1} - \sum_{j=1}^s a_{j,i_1}c_j$ and we can eliminate c_1 in every other block (although we do not know c_1) in the following way: we consider a new vector $\mathbf{p}^{2)} \in \mathbb{F}_q^{ml}$ with components

$$p_i^{2)} = p_i - \frac{a_{1,i}}{a_{1,i_1}}(p_{i_1} - e_{i_1}) = \sum_{j=2}^s a_{j,i}^{2)}c_j + e_i, \text{ for } i \neq i_1,$$

where $a_{j,i}^{2)} = a_{j,i} - \frac{a_{1,i}}{a_{1,i_1}}a_{j,i_1}$, and $p_{i_1}^{2)} = p_{i_1} - e_{i_1}$. Since A is a non-singular by columns matrix, the elements of the first row of A are non-zero, and so the denominator a_{1,i_1} is non-zero.

Since $C_2 \supset \cdots \supset C_s$, we notice that the i-th block of $\mathbf{p}^{2)}$ is a codeword of C_2 plus the error block e_i, for $i \in \{1, \ldots, s\} \setminus \{i_1\}$. We now decode the i_2-th block $p_{i_2}^{2)} = \sum_{j=2}^s a_{j,i_2}^{2)}c_j + e_{i_2}$ of $\mathbf{p}^{2)}$ using LDC_2 and we obtain a list L_2. Since $w(e_{i_2}) \leq \tau_2$, we have $\sum_{j=2}^s a_{j,i_2}^{2)}c_j \in L_2$. In practice we do not know again which one of the elements in L_2 is $\sum_{j=2}^s a_{j,i_2}^{2)}c_j$, therefore we should consider the following computations for every element in L_2. Assume now that we consider $\sum_{j=2}^s a_{j,i_2}^{2)}c_j \in L_2$, hence we obtain e_{i_2} and, as before, we can eliminate c_2 in

every other block (although we do not know c_2) as follows: we consider a new vector $\mathbf{p}^{3)} \in \mathbb{F}_q^{ml}$ with components

$$p_i^{3)} = p_i^{2)} - \frac{a_{2,i}^{2)}}{a_{2,i_2}^{2)}}(p_{i_2}^{2)} - e_{i_2}) = \sum_{j=3}^{s} a_{j,i}^{3)} c_j + e_i, \text{ for } i \neq i_1, i_2,$$

where $a_{j,i}^{3)} = a_{j,i}^{2)} - \frac{a_{2,i}^{2)}}{a_{2,i_2}^{2)}} a_{j,i_2}^{2)}$, $p_{i_1}^{3)} = p_{i_1}^{2)}$ and $p_{i_2}^{3)} = p_{i_2}^{2)} - e_{i_2}$.

Notice that the i-th block of $\mathbf{p}^{3)}$ is a codeword of C_3 plus the error block e_i, for $i \in \{1, \ldots, s\} \setminus \{i_1, i_2\}$.

Then we iterate this process, defining $\mathbf{p}^{k)}$ for $k = 3, \ldots, s$, and decoding the i_k-th block using LDC_k. In this way, we obtain the error blocks e_i, and the corresponding codeword blocks $\sum_{j=1}^{s} a_{j,i} c_j$, for $i \in \{i_1, \ldots, i_s\}$. The vector $(\sum_{j=1}^{s} a_{j,i_1} c_j, \ldots, \sum_{j=1}^{s} a_{j,i_s} c_j)$ formed from these s decoded blocks is equal to the product $[c_1 \cdots c_s] \cdot A(i_1, \ldots, i_s)$, where $A(i_1, \ldots, i_s)$ is the $s \times s$-submatrix of A consisting of the columns $i_1, \ldots, i_s$. Since this matrix is full rank, we can now easily compute $c_1, \ldots, c_s$ by inverting $A(i_1, \ldots, i_s)$ or solving the corresponding linear system. Finally we recover the remaining $l - s$ codeword blocks "for free" (i.e. no decoding procedure is involved for these blocks) by simply recomputing the entire codeword $\mathbf{c} = [c_1 \cdots c_s] \cdot A = (\sum_{j=1}^{s} a_{j,1} c_j, \ldots, \sum_{j=1}^{s} a_{j,l} c_j)$, since we know the c_j's and the matrix A.

For each elimination step in the above procedure, it is necessary that $a_{k,i_k}^{k)} \neq 0$, for each $k = 2, \ldots, s$, to avoid zero division. We claim that this follows from the non-singular by columns property of A, exactly in the same way as in [8]. Let $A^{1)} = A$. The matrix $A^{k)} = (a_{i,j}^{k)}) \in \mathcal{M}(\mathbb{F}_q, s \times l), k = 2, \ldots, s$, is obtained recursively from $A^{k-1)}$ by performing the following $l-(k-1)$ elementary column operations:

$$\text{column}_i(A^{k)}) = \text{column}_i(A^{k-1)}) - \frac{a_{k-1,i}^{k-1)}}{a_{k-1,i_{k-1}}^{k-1)}} \text{column}_{i_{k-1}}(A^{k-1)}),$$

for each $i \notin \{i_1, \ldots, i_{k-1}\}$. These operations introduce $l-(k-1)$ additional zero elements in the $k-1$-th row of $A^{k)}$ at each iteration. Hence the submatrix of $A^{k)}$ given by the first k rows and the $i_1, \ldots, i_k$ columns, is a triangular matrix (in this case, a column permutation of a lower triangular matrix) whose determinant is $a_{1,i_1}^{k)} \cdots a_{k,i_k}^{k)}$. Since A is non-singular by columns, this submatrix is non-singular. It follows that the determinant is non-zero, and therefore $a_{k,i_k}^{k)} \neq 0$.

The procedure described above will generate a list that includes the sent word, if for a given choice of indices $\{i_1, \ldots, i_s\} \subset \{1, \ldots, l\}$, each error block satisfies $wt(e_{i_j}) \leq \tau_j$, for all $j = 1, \ldots, s$. The output's procedure may not include the sent word if $wt(e_{i_j}) > \tau_j$ for some j.

In the previous description, we have only shown the computations for one of the different choices that the list decoder algorithms C_i, for $i = 1, \ldots, s$, may give us. However, if $\#L_1 > 1$ then we should consider a different word $\mathbf{p}^{2)}$ for every $\ell \in L_1$. One can see how this tree is created for every element in L in line 8 of Algorithm 1. If a decoder LDC_j outputs an empty list, for all possible choices in L_{j-1}, then we consider another ordered subset of indices, and start the procedure again.

We now prove that for every error vector $\mathbf{e}$ with $wt(\mathbf{e}) \leq \tau$ there exists a *good* set of indices $\{i_1, \ldots, i_s\} \subset \{1, \ldots, l\}$ satisfying condition (5). We should repeat the procedure described above, with every ordered subset of indices and collect all the decoded words, in order to be sure that the "good" set of indices is considered.

Theorem 3.1. *Let C be the matrix-product code $[C_1 \cdots C_s] \cdot A$, where $C_1 \supset \cdots \supset C_s$ and A is a non-singular by columns matrix. Let $\mathbf{e} = (e_1, e_2, \ldots, e_l) \in \mathbb{F}_q^{ml}$ be an error vector with $wt(\mathbf{e}) \leq \tau$ (see (4)). Then there exists an ordered subset $\{i_1, \ldots, i_s\} \subset \{1, \ldots, l\}$ satisfying $wt(e_{i_j}) \leq \tau_j$, for all $j \in \{1, \ldots, s\}$.*

Proof. We claim that there exists i_1 such that $wt(e_{i_1}) \leq \tau_1$. Suppose that there is no $i_1 \in \{1, \ldots, l\}$ with $wt(e_{i_1}) \leq \tau_1$, that is, $wt(e_i) \geq \tau_1 + 1$, for all $i = 1, \ldots, l$. This implies that

$$wt(\mathbf{e}) = wt(e_1) + \cdots + wt(e_l) \geq l\tau_1 + l > \tau$$

which contradicts our assumption.

Let us assume that the property holds for a subset $\{i_1, \ldots, i_{j-1}\} \subset \{1, \ldots, l\}$ of size $j - 1 < s$. We now prove it holds for a subset of size j. Suppose that there is no i_j with $wt(e_{i_j}) \leq \tau_j$, that is, $wt(e_i) > \tau_j + 1$, for all $i \in \{1, \ldots, l\} \setminus \{i_1, \ldots, i_{j-1}\}$. This implies that

$$\begin{aligned} wt(\mathbf{e}) &\geq \sum_{k=1}^{j-1} wt(e_{i_k}) + \sum_{k=j}^{l} wt(e_{i_k}) \\ &> \sum_{k=1}^{j-1} wt(e_{i_k}) + (l - j + 1)\tau_j + (l - j + 1) \\ &\geq (l - j + 1)\tau_j + (l - j + 1) > \tau \end{aligned}$$

which contradicts our assumption and the result holds. □

Summarizing, we can now formulate our decoding algorithm for $C = [C_1 \cdots C_s] \cdot A \subset \mathbb{F}_q^{ml}$, where $C_1 \supset \cdots \supset C_s$ and A is a non-singular by columns matrix, in procedural form in Algorithm 1.

Corollary 3.2. *If $wt(\mathbf{e}) \leq \tau$ then $\mathbf{c}$ is in the list given as output of Algorithm 1. Hence, the algorithm described in this section is a list decoding algorithm with error bound τ, i.e, $L = \{\mathbf{c} \in C \mid wt(\mathbf{p} - \mathbf{c}) \leq \tau\}$.*

Proof. By 3.1, there exists an ordered subset $\{i_1, \ldots, i_s\} \subset \{1, \ldots, l\}$ satisfying $wt(e_{i_j}) \leq \tau_j$, for all $j \in \{1, \ldots, s\}$. Therefore, $c_{i_j} \in L_j$ and $\mathbf{c} = (\sum_{j=1}^{s} a_{j,1}c_j, \ldots, \sum_{j=1}^{s} a_{j,l}c_j) \in L$. Furthermore, all the words at distance τ from the received word are included in the output list as well. □

Example 3.3. Consider the matrix-product codes with matrix A of the form

$$A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix},$$

and $C_1 \supset C_2$ Reed-Solomon Codes over $\mathbb{F}_{16}$ with parameters $[15, 10, 6]$ and $[15, 4, 12]$, respectively. Therefore, the code $C = [C_1 C_2] \cdot A$ has parameters

Algorithm 1 LIST DECODING ALGORITHM FOR $C = [C_1 \cdots C_s] \cdot A$

Input: Received word $\mathbf{p} = \mathbf{c} + \mathbf{e}$ with $\mathbf{c} \in C$ and $wt(\mathbf{e}) \leq \tau$. $C_1 \supset \cdots \supset C_s$ nested codes and A a non-singular by columns matrix. Decoder LDC_i for code C_i, $i = 1, \ldots, s$.

Output: List of all codewords that differ from p in at most τ places.

1: $\mathbf{p}' = \mathbf{p}$; $A' = A$; $Dec = \{\}$;
2: **for** $\{i_1, \ldots, i_s\} \subset \{1, \ldots, l\}$ **do**
3: $\quad \mathbf{p} = \mathbf{p}'$; $A = A'$; $U' = \{\mathbf{p}\}$;
4: $\quad$ **for** $j = 1, \ldots, s$ **do**
5: $\quad\quad U = U'$; $U' = \{\}$;
6: $\quad\quad$ **for** $\mathbf{u}$ in U **do**
7: $\quad\quad\quad L = LDC_j(u_{i_j})$;
8: $\quad\quad\quad$ **for** ℓ in L **do**
9: $\quad\quad\quad\quad tmp = (0, \ldots, 0) \in \mathbb{F}^m$;
10: $\quad\quad\quad\quad$ **for** $k = j+1, \ldots, s$ **do**
11: $\quad\quad\quad\quad\quad tmp_{i_k} = u_{i_k} - \frac{a_{j,i_k}}{a_{j,i_j}}\ell$;
12: $\quad\quad\quad\quad$ **end for**
13: $\quad\quad\quad\quad U' = U' \cup \{tmp\}$;
14: $\quad\quad\quad$ **end for**
15: $\quad\quad$ **end for**
16: $\quad\quad$ **if** $U' = \{\}$ **then**
17: $\quad\quad\quad$ *Break* the loop and consider another $i_1, \ldots, i_s$ in line 2;
18: $\quad\quad$ **end if**
19: $\quad\quad$ **for** $k = j+1, \ldots, s$ **do**
20: $\quad\quad\quad \text{column}_{i_k}(A) = \text{column}_{i_k}(A) - \frac{a_{j,i_k}}{a_{j,i_j}}\text{column}_{i_j}(A)$;
21: $\quad\quad$ **end for**
22: $\quad$ **end for**
23: $\quad$ **for** $\mathbf{u}$ in U' **do**
24: $\quad\quad$ Obtain $(c_1, \ldots, c_s)$ from $u_{i_1}, \ldots, u_{i_s}$;
25: $\quad\quad \mathbf{p} = [c_1 \cdots c_s] \cdot A$; (see (1) and (2))
26: $\quad\quad$ **if** $wt(\mathbf{p} - \mathbf{p}') \leq \tau$ **then**
27: $\quad\quad\quad Dec = Dec \cup \{\mathbf{p}\}$;
28: $\quad\quad$ **end if**
29: $\quad$ **end for**
30: **end for**

[30, 14, 12]. We have error bounds $\tau_1 = 3, \tau_2 = 7$, for C_1 and C_2 respectively, using the list decoding algorithm in [1] or in [13] with multiplicity $v = 4$ (see next section for further details). Therefore, the error bound for Algorithm 1 is $\tau = 7$. Note that the error correction capability of C with [8] is only $t = 5$.

Let $\mathbf{c} = (0, 0)$ be the sent word and $\mathbf{p} = (\alpha^2 x + \alpha x^5 + \alpha^5 x^6 + \alpha^{14} x^{13}, \alpha^5 x^2 + \alpha^7 x^6 + \alpha^8 x^{10})$ the received word, i.e. $wt(\mathbf{e}) = 7$.

- We consider the ordered set of indices $\{1, 2\}$. So, we decode $p_1 = \alpha^2 x + \alpha x^5 + \alpha^5 x^6 + \alpha^{14} x^{13}$ with the list decoding algorithm for C_1: we obtain $p_1^{2)} = \alpha^2 x + \alpha x^5 + \alpha^5 x^6 + \alpha^{14} x^7 + \alpha^{10} x^{13} + \alpha^5 x^{14}$.

  Then we compute $p_2^{2)} = p_2 - p_1^{2)} = \alpha^2 x + \alpha^5 x^2 + \alpha x^5 + \alpha^5 x^6 + \alpha x^7 +$

$\alpha^8 x^{10} + \alpha^{10} x^{13} + \alpha^5 x^{14}$ and decode it with the list decoding algorithm for C_2. However, we get an empty list as output and we do not consider any codeword for the final list.

- We consider now the ordered set of indices $\{2, 1\}$. Therefore, we decode $p_2 = \alpha^5 x^2 + \alpha^7 x^6 + \alpha^8 x^{10}$ with the list decoding algorithm for C_1. We obtain as output $p_2^{2)} = 0$. Thus, we compute $p_1 - p_2^{2)} = p_1$ and we decode it with the list decoding algorithm for C_2. We have obtain 0. Therefore, we deduce that the sent codeword is $(0, 0)$.

Remark 3.4. Matrix-product codes are generalized concatenated codes [2]. There is an efficient decoding algorithm for generalized concatenated codes [4], the cascaded decoding of multilevel concatenations. In [6], Guruswami and Rudra generalize this algorithm to a list decoding algorithm for generalized concatenated codes.

The latter are defined as follows: consider s outer codes, say C_{out}^j over $\mathbb{F}_{q^{a_j}}$ with parameters (N, K_j, D_j) for $j = 0, \ldots, s-1$. Let $C_{out} = C_{out}^0 \times \cdots \times C_{out}^{s-1} = \{(c^0, \ldots, c^{s-1}) \mid c^j \in C_{out}^j, j = 0, \ldots, s-1\}$, understanding $c^j \in C_{out}^j$ as a row vector. Thus, a typical element $c \in C_{out}$ is a $s \times N$ matrix, we denote by c_k the k-th column of c, for $k = 0, \ldots, N-1$.

One also considers a inner code C_{in} over $\mathbb{F}_q$ and a one-to-one map ψ from $\mathbb{F}_{q^{a_0}} \times \cdots \times \mathbb{F}_{q^{a_{s-1}}}$ to C_{in} that maps $(i_0, \ldots, i_{s-1}) \in \mathbb{F}_{q^{a_0}} \times \cdots \times \mathbb{F}_{q^{a_{s-1}}}$ to a codeword $\psi(i_0, \ldots, i_{s-1})$ in C_{in}.

A generalized concatenated code V of order s is the set

$$V = \{\psi(c_0), \ldots, \psi(c_{N-1}) \mid (c_0, \ldots, c_{N-1}) \in B\}.$$

Guruswami and Rudras's algorithm [6] works as follows (we do not attempt to write the algorithm, just to describe its main idea): let $R \in \mathcal{M}(a_0 + \cdots + a_{s-1} \times N, \mathbb{F}_q)$ be the received word. For $j = 1, \ldots, s-1$ consider the code C_{in}^j generated by all the rows of the generator matrix in C_{in} except the first $a_0 + \cdots + a_{j-1}$, say G_{in}^j, where $C_{in}^0 = C_{in}$. The algorithm assumes the existence of list decodable algorithms for C_{in}^j and list recovery algorithm for C_{out}^j. It is also assumed that the list decoding algorithm for C_{in}^j returns a list of messages while the list recovery algorithm for C_{out}^j returns a list of codewords.

In the first round one applies, for $i = 0, \ldots, N-1$, a list decoding algorithm of C_{in}^0 to $\psi(c_i)$ obtaining a list S_i^0. For $i = 0, \ldots, N-1$ and for each element in S_i^0, one can recover a message $\tilde{c_i}$ and project it into the first component, obtaining a list T_i^0. Then it applies list recovery algorithm of C_{out}^0 to $\{T_i^0\}_i$ obtaining a list L_0 which in particular contains c^0.

In the second round we proceed as follows, for each $c \in L_0$, $c \in \mathcal{M}(a_0 \times N, \mathbb{F}_q)$, consider the matrix $\hat{c} \in \mathcal{M}(a_0 + \cdots + a_{s-1} \times N, \mathbb{F}_q)$ which is the matrix of zeroes with c in the first a_0 rows. One may subtract $R = R - (G_{in}^0)^t \hat{c}$, i.e., if $c = c^0$ we are cancelling it from the received word. Thus we have a new received word R that should be decode with the generalized concatenated code with outer codes $C_{out}^1, \ldots, C_{out}^{s-1}$ and inner code C_{in}^1. Since the number of outer codes has dropped by one, repeating this process s-times one can successfully list decode the original generalized concatenated code.

Let C_{in} be a linear code over $\mathbb{F}_q$ with parameters $[n, k, d]$ and generator matrix A, and let C_{out}^j be a linear code over $\mathbb{F}_q$ with parameters $[N, K_j, D_j]$ for $j =$

$0, \ldots, k-1$. Consider ψ the encoding linear map of C_{in}, i.e., $\psi(i_0, \ldots, i_{k-1}) = (i_0, \ldots, i_{k-1})A$. Then the generalized concatenated code V is equal to the matrix product code $[C^1_{out}, \ldots, C^s_{out}] \cdot A$. Since a matrix-product code $[C_1, \ldots, C_s] \cdot A$, with A non-singular by columns, is a generalized concatenated code one might use the algorithm in [6] for matrix-product codes as well. However, the algorithm in [6] can not be successfully applied for matrix-product codes because the inner code has generator matrix A, that is a small matrix (in practice).

Theorem 3.1 guarantees the existence of a *good* set of indices, i.e., satisfying $wt(e_{i_j}) \leq \tau_j$, for all $j \in \{1, \ldots, s\}$. Hence, in the worst case, we may have to consider $s!\binom{\ell}{s}$ iterations. However, in average we will consider much fewer iterations. Given a fix set of ordered indices $S = \{i_1, \ldots, i_s\}$, we will estimate for how many error patterns of weight τ, one has a *good* set of indices. In other words, what is the probability that a set of indices $\{i_1, \ldots, i_s\}$ is a *good* in the worst case, i.e. when τ errors occur.

Proposition 3.5. *If τ errors occur the probability that a fix set of indices $\{i_1, \ldots, i_s\}$ verifies that $wt(e_{i_j}) \leq \tau_j$ for all $j \in \{1, \ldots, s\}$ is:*

$$\frac{\sum_{a_1=0}^{\tau_1} \sum_{a_2=0}^{min\{\tau-a_1,\tau_2\}} \cdots \sum_{a_{s-1}=0}^{min\{\tau-\sum_{j=1}^{s-2} a_j, \tau_{s-1}\}} \binom{m}{a_1}\binom{m}{a_2}\cdots\binom{m}{a_{s-1}}\binom{m(\ell-s+1)}{\tau-\sum_{j=1}^{s-1} a_j}}{\binom{m\ell}{\tau}}$$

Proof. Let us compute the different error vectors $\mathbf{e}$ with weight τ that allow us to have a *good* set of indices. If $wt(e_{i_1}) \leq \tau_1$, then we may have $0 \leq a_1 \leq \tau_1$ errors in block i_1, and there are $\binom{m}{a_1}$ possibilities for having a_1 errors in block i_1. Assuming that a_1 errors occurred in the block i_1, we may have $wt(e_{i_2}) \leq \tau_2$ if and only if there are a_2 errors in the block i_2, where $0 \leq a_2 \leq min\{\tau - a_1, \tau_2\}$, since $wt(\mathbf{e}) = \tau$. Hence, there are $\binom{m}{a_2}$ different possibilities for having a_2 errors in the second block. Repeating this argument for the first $s-1$ blocks, we may have $\tau - \sum_{j=1}^{s-1} a_j$ errors in the $\ell - s + 1$ remaining blocks (including block i_s) and therefore there are $\binom{m(\ell-s+1)}{\tau-\sum_{j=1}^{s-1} a_j}$ possibilities for the remaining blocks. Overall we have

$$\sum_{a_1=0}^{\tau_1} \sum_{a_2=0}^{min\{\tau-a_1,\tau_2\}} \cdots \sum_{a_{s-1}=0}^{min\{\tau-\sum_{j=1}^{s-2} a_j, \tau_{s-1}\}} \binom{m}{a_1}\binom{m}{a_2}\cdots\binom{m}{a_{s-1}}\binom{m(\ell-s+1)}{\tau-\sum_{j=1}^{s-1} a_j}$$

error patterns that make $\{i_1, \ldots, i_s\}$ a *good* set of indices. The result holds since there are $\binom{m\ell}{\tau}$ error vectors of weight τ. □

In Example 3.3 we have $s = l = 2$ and $\tau_1 = 3$. Therefore the probability that either $\{1, 2\}$ or $\{2, 1\}$ is a good set of indices is

$$\frac{\binom{15}{0}\binom{15}{7} + \binom{15}{1}\binom{15}{6} + \binom{15}{2}\binom{15}{5} + \binom{15}{3}\binom{15}{4}}{\binom{30}{7}} = \frac{1}{2}.$$

Finally we consider the complexity of alogithm 1.

Theorem 3.6. *Let $LDC_1, \ldots, LDC_s$ be the list decoding algorithms considered in Algorithm 1 with error bounds $\tau_1, \ldots, \tau_s$, respectively, and that output*

a list with size bounded by $D_1, \ldots, D_s$. We denote by R_i the complexity of the algorithm LDC_i. Then, algorithm 1 has complexity

$$O\left(s!\binom{\ell}{s}(D_1 + \sum_{i=2}^{s}(\prod_{j=1}^{i-1} D_j)R_i)\right).$$

Proof. In the worst case, one should consider every ordered set of s elements within the l possible indices, that is $s!\binom{\ell}{s}$ ordered sets. For a fix ordered set, we run LDC_1 which yields a list of size at most D_1, in the worst case. For each element in this list we run LDC_2 producing a list of size D_2, in the worst case. Hence we will have $D_1 D_2$ words that should be decoded with LDC_3. Repeating this process, in the worst case, we will decode $D_1 \cdots D_{s-1}$ words with LDC_s. □

4 List decoding of Matrix-product codes from Reed-Solomon codes with small s

The previous algorithm can become computationally intensive as the number of blocks l, the number of blocks that we may need to decode at each iteration s and the error bounds $\tau_1, \ldots, \tau_s$, increase. Therefore, there are two interesting situations: considering few blocks and considering codes and error bounds such that there is a small probability of getting a list with more that one element as output of the list decoding algorithms LDC_i.

Let us consider that the constituent codes $C_1 \supset \cdots \supset C_s$ are Reed-Solomon. This family of codes is especially interesting in this setting for two reasons, consider an $[m, k, d]$ Reed-Solomon code, there is an efficient list-decoding algorithm [7] for decoding up to τ^v errors, with multiplicity $v \in \mathbb{N}$, which is computed as follows

$$\tau^v = m - \left\lfloor \frac{l_v}{v} \right\rfloor - 1, \text{ where}$$

$$l_v = \left\lfloor \frac{m\binom{v+1}{2}}{r_v} + \frac{(r_v - 1)(k-1)}{2} \right\rfloor \text{ and } r_v \text{ is calculated so that}$$

$$\binom{r_v}{2} \leq \frac{m\binom{v+1}{2}}{k-1} < \binom{r_v + 1}{2}.$$

In particular, one has the algorithm in [13] with complexity $O(D^4 v m^2)$ and the one in [1] with complexity $O(D^4 v m \log^2 m \log\log m)$, where D is the list size, v the multiplicity and m the length of the code. Notice that, by fixing v, we fix τ^v and bound the list size $D \leq l_v/(k-1)$. Thus, one may obtain the complexity of algorithm 1 as a function of the multiplicities and the length of the constituent Reed-Solomon codes by Theorem 3.6.

Furthermore, one has a bound for the the probability $p_{\tau^v}(C)$ that the output of Guruswami-Sudan's algorithm for the code C with error bound τ^v has more than 1 codeword [14], given that at most τ^v errors have occurred and assuming that all the error patterns have the same probability. It turns out that this probability may be very small in practice, for example a Reed-Solomon code over $\mathbb{F}_{2^6}$ with parameters $[64, 20, 45]$ and $\tau^1 = 23$, this probability is 10^{-25}.

We consider codes and error bounds in such a way that $p_{\tau_i}(C_i)$ is small, we abbreviate $\tau_i^{v_i}$ to τ_i. With the notation of the previous section, consider a received word $\mathbf{p} = \mathbf{c} + \mathbf{e}$ where $\mathbf{e}$ is the error vector with $wt(\mathbf{e}) \leq \tau$. Consider the ordered set of indices $\{i_1, \ldots, i_s\} \subset \{1, \ldots, l\}$, if $wt(e_{i_j}) \leq \tau_j$ for every j then we say that the set of indices is *good* (otherwise we say that it is *bad*). For a *good* set of of indices, the sent word $\mathbf{c}$ is in the output list by Theorem 3.1. Furthermore, we claim that with a high probability the output list just contains this word: $LDC_1(p_{i_1})$ is going to give as output a list containing $\sum_{j=1}^{s} a_{j,1}c_j$, in Algorithm 1. In practice this list will only have one element since the probability of getting just one codeword is $1 - p_{\tau_1}(C_1)$. Then we eliminate c_1 in the block i_2 and decode it using LDC_2. Since $w(e_{i_2}) \leq \tau_2$ we obtain a list that contains $\sum_{j=2}^{s} a_{j,i_2}^{2)} c_j$ and with high probability this list has only one codeword. We proceed in the same way for the rest of the blocks and with probability

$$\prod_{i=1}^{s}(1 - p_{\tau_i}(C_i)) \tag{6}$$

we obtain an output with just one codeword for this set of indices.

Consider now a *bad* set of indices $\{i_1, \ldots, i_s\}$, that is, there exists j such that $w(e_{i_1}) < \tau_1, \ldots, w(e_{i_{j-1}}) < \tau_{j-1}$, but $w(e_{i_j}) < \tau_j$, then the block j will not be correctly decoded. Again, with probability (6) we will obtain at most one codeword $\mathbf{p}'$ for this set of indices, we do not know anything about this codeword excepting that it is not the sent one. However, we claim that with a high probability the codewords obtained with this *bad* set of indices will be discarded in line 26 of Algorithm 1, namely, we claim that $wt(\mathbf{p} - \mathbf{p}') > \tau$ with at least probability $1 - lp_{\tau_1}(C_1)$.

Lemma 4.1. *Let $\mathbf{p}, \mathbf{p}' \in C = [C_1 \cdots C_s] \cdot A$, with $\mathbf{p} \neq \mathbf{p}'$ and $C_1 \supset \cdots \supset C_s$ Reed-Solomon codes. For τ as in (4), we have*

$$P(wt(\mathbf{p} - \mathbf{p}') < \tau) \leq lp_{\tau_1}(C_1).$$

Proof. If $wt(\mathbf{p} - \mathbf{p}') \leq \tau$ then there is j such that

$$wt(p_j - p'_j) \leq \tau/l \leq \tau/s \leq \tau_1.$$

One has that $P(wt(p_i - p'_i) < \tau_1) = p_{\tau_1}(C_1)$, since $\mathbf{p} \neq \mathbf{p}'$ and $p_i, p'_i \in C_1$. Thus,

$$P(wt(\mathbf{p} - \mathbf{p}') < \tau) \leq \sum_{i=1}^{l} P(wt(p_i - p'_i) < \tau_1) = lp_{\tau_1}(C_1)$$

since $\mathbf{p} \neq \mathbf{p}'$ and $wt(\mathbf{p} - \mathbf{p}') = \sum_{i=1}^{l} wt(p_i - p'_i)$. □

An optimal situation arises considering $s = l = 2$ and two Reed-Solomon codes $C_1 \supset C_2$ such that for error bounds τ_1 and τ_2, respectively, Guruswami-Sudan's Algorithm outputs a list of at most 1 element with a high probability (if at most τ_1, τ_2, respectively, errors have occurred). When $l = s = 2$, this construction gives the same family of codes as the $(u, u + v)$-construction, for instance Reed-Muller codes are obtained in that way.

5 Bounded Distance Decoding of Quasi-Cyclic Codes

Let $C_1, \ldots, C_s \subset \mathbb{F}_q^m$ be cyclic codes of length m and $A = (a_{i,j})$ an $s \times l$-matrix, with $s \leq l$, whose entries are units in the ring $\mathbb{F}_q[x]/(x^m - 1)$ or zero. A unit in $\mathbb{F}_q[x]/(x^m - 1)$ is a polynomial of degree lower than m whose greatest common divisor with $x^m - 1$ is 1. The so-called *matrix-product code with polynomial units* is the set $C = [C_1 \cdots C_s] \cdot A$ of all matrix-products $[c_1 \cdots c_s] \cdot A$ where $c_i \in C_i \subset \mathbb{F}_q[x]/(x^m - 1)$ for $i = 1, \ldots, s$. These codes were introduced in [9].

We consider always a special set of matrices A to be defined below with full-rank over $\mathbb{F}_q[x]/(x^m - 1)$. Let C_i with parameters $[m, k_i, d_i]$, then the matrix-product code with polynomial units $C = [C_1 \cdots C_s] \cdot A$ has length lm and dimension $k = k_1 + \cdots + k_s$.

Let $R_i = (a_{i,1}, \ldots, a_{i,l})$ be the element of $(\mathbb{F}_q[x]/(x^m - 1))^l$ consisting of the i-th row of A, where $i = 1, \ldots, s$. Let C_{R_i}, be the $\mathbb{F}_q[x]/(x^m - 1)$-submodule of $(\mathbb{F}_q[x]/(x^m - 1))^l$ generated by $R_1, \ldots, R_i$. In other words, C_{R_i} is a linear code over a ring, and we denote by D_i the minimum Hamming weight of the words of C_{R_i}, $D_i = \min\{wt(x) \mid x \in C_{R_i}\}$. In [9] the following bound on the minimum distance was obtained

$$d(C) \geq d^* = \min\{d_1 D_1, d_2 D_2, \ldots, d_s D_s\}. \tag{7}$$

One of the differences between matrix-product codes and matrix-product codes with polynomial units is that the lower bound d^* is not sharp for the latter class of codes.

The minimum distance can actually be much larger than d^* and several codes with very good parameters were obtained in this way in [9]. We will provide a bounded distance decoding algorithm for these codes, using the list-decoding Algorithm 1.

Matrix-product codes with polynomial units are quasi-cyclic codes [12] of length ml. Although this family provides codes with very good parameters there are no general fast algorithms for decoding them. The algorithm in [8] for matrix-product codes may be used for these family of codes under certain hypothesis, but in that case it only corrects up to $\lfloor \frac{d^*-1}{2} \rfloor$.

We remark that the units of a ring form a multiplicative group, however they do not form an additive group. That is, if $f, g \in \mathbb{F}_q[x]/(x^m - 1)$ are units, then fg is a unit but $f + g$ or $f - g$ are not a unit in general. This phenomena will impose further restrictions for the list-decoding algorithm since we cannot divide by a non-unit.

Definition 5.1. Let A be a $s \times l$ matrix, whose entries are units in the ring $\mathbb{F}_q[x]/(x^m - 1)$ or zero. Let A_t be the matrix consisting of the first t rows of A. For $1 \leq j_1 < \cdots < j_t \leq l$, we denote by $A(j_1, \ldots, j_t)$ the $t \times t$ matrix consisting of the columns $j_1, \ldots, j_t$ of A_t.

A matrix A is **unit by columns** if the determinant of $A(j_1, \ldots, j_t)$ is a unit in $\mathbb{F}_q[x]/(x^m - 1)$ for each $1 \leq t \leq s$ and $1 \leq j_1 < \cdots < j_t \leq l$.

In particular, a unit by column matrix is a non-singular by columns matrix.

Let C be a matrix-product code with polynomial units i.e. $C = [C_1 \cdots C_s] \cdot A$, where Let $C_1 \supset \cdots \supset C_s$ and A is a unit by columns matrix, in particular the

elements of the first row of A are non-zero.

With the notation of section 3, consider a received word $\mathbf{p} = \mathbf{c} + \mathbf{e}$ where $\mathbf{e}$ is the error vector with $wt(\mathbf{e}) \leq \tau$.

For $s = 1$, the definitions of non-singular by column- and unit by column - matrix are the same. Namely, we can use Algorithm 1 without any modifications: for $\{i_1\} \subset \{1, \ldots, l\}$ a *good* set of indices, we decode the block p_{i_1} with LDC_1 because the cyclic codes generated by f and by fu, with $f \mid x^m - 1$ and $\gcd(u, x^m - 1) = 1$, are the same code. Then we divide by a_{1,i_1} to recover c_1 (in line 24), we can consider the inverse of a_{1,i_1} since the entries of A are units. Actually, this algorithm for $s = 1$ is the list decoding version of the algorithm in [11] for 1-generator 1-level quasi-cyclic codes.

For $s \geq 2$, for each elimination step in Algorithm 1, we are dividing by a_{j,i_j} (in lines 11, 20), we claim that this can be performed because a_{j,i_j} is a unit. Let $A^{k)}$ denote the matrix obtained recursively from A by performing the following $l - (k-1)$ elementary column operations (see section 3):

$$\text{column}_i(A^{k)}) = \text{column}_i(A^{k-1)}) - \frac{a^{k-1)}_{k-1,i}}{a^{k-1)}_{k-1,i_{k-1}}} \text{column}_{i_{k-1}}(A^{k-1)}),$$

for each $i \notin \{i_1, \ldots, i_{k-1}\}$. These operations introduce $l-(k-1)$ additional zero elements in the $k-1$-th row of $A^{k)}$ at each iteration. Hence the submatrix of $A^{k)}$ given by the first k rows and the $i_1, \ldots, i_k$ columns, is a triangular matrix (in this case, a column permutation of a lower triangular matrix) whose determinant is $a^{k)}_{1,i_1} \cdots a^{k)}_{k,i_k}$. Since A is unit by columns, this minor is a unit. Hence, $a^{k)}_{k,i_k}$ is a unit, since the units form a multiplicative group.

Thus, we have a list-decoding algorithm with error bound τ as in (4). Furthermore, we can use it also for unique decoding up to the capacity of the code if $\tau = \lfloor \frac{d(C)-1}{2} \rfloor$.

Theorem 5.2. *Consider a matrix-product code with polynomial units $C = [C_1 \cdots C_s] \cdot A$, where $C_1 \supset \cdots \supset C_s$ and A is a unit by columns matrix. Let $\tau = \lfloor \frac{d(C)-1}{2} \rfloor$, then the list decoding Algorithm 1 is a unique decoding algorithm for C.*

Proof. We have seen above that Algorithm 1 can be successfully applied in this setting. Hence, this algorithm is a list decoding algorithm, by Corollary 3.2. In particular, if $wt(\mathbf{e}) \leq \tau$ the sent word is in the output list. Moreover, since $\tau = \lfloor \frac{d(C)-1}{2} \rfloor$ there is no other codeword at distance τ from the received word and the result holds. □

Example 5.3. Let $s = 1$, $l = 2$, and let C_1 be the Reed-Solomon code with parameters $[15, 8, 8]$ and generator polynomial $f = x^7 + \alpha^6 x^6 + \alpha^{13} x^5 + \alpha^{12} x^4 + \alpha x^3 + \alpha^{10} x^2 + \alpha^{11} x + \alpha^{13}$, where α is a primitive root $\mathbb{F}_{16}$. Let $C = [C_1] \cdot A$, where $A = [1, x^4 + \alpha^5 x^3 + \alpha x^2 + \alpha^{11} x + \alpha^{14}]$, with $\alpha \in \mathbb{F}_{16}$ a primitive element. One has that C is a quasi-cyclic code with parameters $[30, 8, 19]$.

The error correction capability of C is $t = 9$. However, with the algorithm in [8] we can only decode up to 7 errors, since $d^* = 16$. Considering a list decoding algorithm with multiplicity 2 for C_1, we have an error bound $\tau_1 = 4$. Hence the error correction capability of Algorithm 1 is $\tau = 2\tau_1 + 1 = 9$ and we have that it is a unique decoding algorithm for C.

Example 5.4. Let $s = 1$, $l = 2$, and let C_1 be the Reed-Solomon code with parameters $[15, 5, 11]$ and generator polynomial $f = x^{10} + \alpha^2 x^9 + \alpha^3 x^8 + \alpha^9 x^7 + \alpha^6 x^6 + \alpha^{14} x^5 + \alpha^2 x^4 + \alpha x^3 + \alpha^6 x^2 + \alpha x + \alpha^{10}$, where α is a primitive element in $\mathbb{F}_{16}$. Let $C = [C_1] \cdot A$, where $A = [1, x^3 + \alpha^3 x^2 + \alpha^{14} x + \alpha^9]$. One has that C is a quasi-cyclic code with parameters $[30, 5, 24]$, which is the best known code in [3].

The error correction capability of C is $t = 11$, however, with the algorithm in [8] we can only decode up to 10 errors, since $d^* = 22$. Considering a list decoding algorithm for C_1 with multiplicity $v = 1$, we have error bound $\tau_1 = 5$ and Algorithm 1 decodes up to the half of the minimum distance since its correction capability is $\tau = 2\tau_1 + 1 = 11$. However, considering a list decoding algorithm for C_1 with multiplicity $v = 8$, we have error bound $\tau_1 = 7$. Hence, the error bound for Algorithm 1 is $\tau = 2\tau_1 + 1 = 15$, which is a list decoding algorithm for C.

Example 5.5. Let $s = l = 2$ and consider a matrix A of the form

$$A = \begin{pmatrix} 1 & g \\ 0 & 1 \end{pmatrix},$$

with g a unit in $\mathbb{F}_2[x]/(x^m - 1)$. One has that A is a unit by column matrix.

Consider $C_1 \supset C_2$ Reed-Solomon Codes over $\mathbb{F}_{16}$ with parameters $[15, 13, 3]$ and $[15, 8, 8]$, respectively. We consider the unit $g = x^5 + \alpha^{10} x^3 + \alpha^2 x^2 + \alpha^2$. One has that the code $C = [C_1 C_2] \cdot A$ has parameters $[30, 21, 7]$. Hence, its error correction capability is $t = 3$. Let $\tau_1 = 1, \tau_2 = 3$ be the error bounds for C_1 and C_2, for list decoding algorithms with multiplicity 1. Thus, the error bound for Algorithm 1 is $\tau = 3$ and we have a unique decoding algorithm for C.

Note that we may consider unique decoding algorithms for C_1 and C_2 since $\tau_i = \lfloor \frac{d_i - 1}{2} \rfloor$, for $i = 1, 2$. Hence, in this case, we can reduce the complexity of the algorithm by considering unique decoding algorithms.

6 Conclusion

In this article we described a list-decoding algorithm for a class of Matrix-Product codes, we computed its error bound and complexity. This algorithm can become computationally intense, however we show that for small s, ℓ and considering Reed-Solomon codes as constituent codes, the algorithm does not become computationally intense. Furthermore, we are able to bound the probability of getting more than one codeword as output. The main advantage of this approach with respect to Reed-Solomon codes is the possibility of considering longer codes without increasing the field size and still using the fast decoding algorithms [1, 13] for the constituent codes. Moreover, we can consider a bounded distance decoding, that decodes up to half of the minimum distance, for Matrix-Product codes with polynomial units, a family with very good parameters.

References

[1] Peter Beelen and Kristian Brander. Key equations for list decoding of Reed-Solomon codes and how to solve them. *J. Symbolic Comput.*, 45(7):773–786,

2010.

[2] Tim Blackmore and Graham H. Norton. Matrix-product codes over $\mathbb{F}_q$. *Appl. Algebra Engrg. Comm. Comput.*, 12(6):477–500, 2001.

[3] Schmid Wolfgang Ch. and Schrer Rudolf. Mint. *Dept. of Mathematics, University of Salzburg*, http://mint.sbg.ac.at/about.php.

[4] Ilya I. Dumer. Concatenated codes and their multilevel generalizations. In *Handbook of coding theory, Vol. I, II*, pages 1911–1988. North-Holland, Amsterdam, 1998.

[5] Peter Elias. *List decoding for noisy channels.* Research Laboratory of Electronics, Massachusetts Institute of Technology, Cambridge, Mass., Rep. No. 335, 1957.

[6] Venkatesan Guruswami and Atri Rudra. Better binary list decodable codes via multilevel concatenation. *IEEE Trans. Inform. Theory*, 55(1):19–26, 2009.

[7] Venkatesan Guruswami and Madhu Sudan. Improved decoding of Reed-Solomon and algebraic-geometry codes. *IEEE Trans. Inform. Theory*, 45(6):1757–1767, 1999.

[8] Fernando Hernando, Kristine Lally, and Diego Ruano. Construction and decoding of matrix-product codes from nested codes. *Appl. Algebra Engrg. Comm. Comput.*, 20(5-6):497–507, 2009.

[9] Fernando Hernando and Diego Ruano. New linear codes from matrix-product codes with polynomial units. *Adv. Math. Commun.*, 4(3):363–367, 2010.

[10] T. Kasami. A Gilbert-Varshamov bound for quasi-cyclic codes of rate 1/2. *IEEE Trans. Information Theory*, IT-20:679, 1974.

[11] K. Lally. Quasicyclic codes - some practical issues. In *Proceedings. 2002 IEEE International Symposium on Information Theory*, 2002.

[12] Kristine Lally and Patrick Fitzpatrick. Algebraic structure of quasicyclic codes. *Discrete Appl. Math.*, 111(1-2):157–175, 2001.

[13] Kwankyu Lee and Michael E. O'Sullivan. List decoding of Reed-Solomon codes from a Gröbner basis perspective. *J. Symbolic Comput.*, 43(9):645–658, 2008.

[14] R. Refslund Nielsen and T. Høholdt. Decoding Reed-Solomon codes beyond half the minimum distance. In *Coding theory, cryptography and related areas (Guanajuato, 1998)*, pages 221–236. Springer, Berlin, 2000.

[15] Ferruh Özbudak and Henning Stichtenoth. Note on Niederreiter-Xing's propagation rule for linear codes. *Appl. Algebra Engrg. Comm. Comput.*, 13(1):53–56, 2002.

[16] John M. Wozencraft. List decoding. In *Quarterly Progress Report*, pages 90–95. Cambridge, MA:Res. Lab. Electronics, MIT, 1958.