Linear Form Decoding - Decoding Binary Codes on Non-binary Channels

Harm S. Cronie, Bertrand Ndzana Ndzana and Amin Shokrollahi.

Abstract

In this paper, we introduce a class of decoding algorithms for binary codes used for transmission over q-ary channels, where $q = 2^m$. We apply the algorithms to codes that can be decoded by belief propagation algorithms. The algorithms provide a tradeoff between complexity and decoding capability. Whereas the running time of the q-ary belief-propagation algorithm is $m2^m$ times that of its binary counterpart, in our case the complexity factor can be chosen between m and 2^m, depending on the error-correction capability required. As such, the running time can be much better than the q-ary belief-propagation algorithm. The main idea behind our algorithm is to apply an appropriate set of $\mathbb{F}_2$-linear forms of $\mathbb{F}_q$ to a code to obtain a set of binary codes which can be decoded independently in parallel. Once each of these binary codes have been decoded we decode the code defined by the set of $\mathbb{F}_2$-linear forms. Simulations results are provided for families of q-ary symmetric channels which show the performance of our decoding algorithms.

I. Introduction

The success of iterative decoding algorithms for graph based codes on binary memoryless channels has for some time invited the question whether it is possible to generalize binary decoding algorithms and tools for their design and analysis to q-ary alphabets. Such generalizations are needed in a variety of applications. For instance many channels in communication applications

Harm S. Cronie is with the EPFL, Lausanne, Switzerland and is supported by Shokrollahi's Grant 228021-ECCSciEng of the European Research Council.

Bertrand Ndzana Ndzana was with the EPFL, Lausanne, Switzerland and was supported by Shokrollahi's Grant 228021-ECCSciEng of the European Research Council.

Amin Shokrollahi is with the EPFL, Lausanne, Switzerland and is supported by Grant 228021-ECCSciEng of the European Research Council.

are non-binary in nature. These channels require higher modulation than binary. In this case it is natural to opt for a non-binary error-correcting code. Another application where non-binary error-correcting codes are natural to use is the compression of files with error-correcting codes. In this case the alphabet elements are naturally bytes.

One often would like to use binary error-correcting codes since they can be decoded with less computational complexity than their non-binary counterparts. For some channels schemes exist which allow the use of binary codes on non-binary channels without much penalty. One can e.g. opt for multi-level coding [1], [2] where a set of binary codes is used and decoded in a sequential fashion. The main disadvantage of multi-level coding is error propagation which limits the applicability of the scheme when the alphabet size increases and block lengths are short. A successful scheme for channels arising in communication applications is bit-interleaved coded-modulation (BICM) [3]. This scheme achieves a good practical performance on channels for which the parallel-and-independent decoding (PID) capacity is close to the capacity of the channel. Furthermore, there is no issue of error-propagation. However, BICM is not suitable for any non-binary channel since there can be a large gap between the capacity of the channel and its PID capacity.

On the other hand, it is relatively straightforward to generalize the binary belief-propagation (BP) algorithm to the q-ary case [4]: assuming that the underlying alphabet is $\mathbb{F}_q$, the messages that are passed along the edges of the graph are probability distributions on $\mathbb{F}_q$, and can be described by a vector of $(q-1)$ nonnegative real numbers with sum at most equal to 1. The message passing algorithm performs convolutions of these probability distributions on one side of the graph, and point-wise products (and renormalization) on the other side. The convolution can be performed at cost proportional to $q \log_2(q)$ using a Hadamard-Walsh transform [5], [6]. For a tutorial describing existing methods for decoding q-ary graphical codes, and extensions of these algorithms, we refer the reader to [7].

Despite improvements reported in [7], the problem of decoding graphical codes over $\mathbb{F}_q$ remains a difficult computational problem, particularly when the field size q increases. Belief-propagation based methods are rendered completely ineffective when the field size q is very large, as is for example the case in packet based transmissions where q can have a value of 2^{8192} or more. In these cases, under the assumption that transmission is performed on the q-ary symmetric channel (q-SC), several algorithms have been provided [8], [9], [10]. However, these

algorithms are ineffective when q is not too large, say in the range of a few hundreds to a few thousands.

In this paper, we are going to ask a slightly different question: to what extent can we use binary codes on non-binary channels, and how can we design efficient decoding algorithms? It is quite clear that if the q-ary channel induces independent errors on the bits of a transmitted element, then the error correction capability of binary codes is similar to that of binary codes on binary channels. However, if the channel does not introduce independent errors at the bit-level (something that is common for q-ary channels), then binary codes could have an advantage.

In this paper we introduce a technique to which we refer to as *linear form decoding*. Part of our results presented in this paper have been presented in [11]. The main idea of linear form decoding is to reduce the decoding problem of a set of m binary codes, which will be used on a non-binary channel, to multiple correlated instantiations of the decoding problem of the binary codes. The number of correlated instantiations that is used can be chosen anywhere between m and 2^m. Furthermore, the number of correlated instantiations used, gives a trade-off between complexity and performance. Once the correlated instantiations have been decoded, we combine the results to estimate the transmitted message.

The outline of the paper is as follows. In Section II we explain the main concepts of linear form decoding. In Section III we illustrate the capabilities of linear form decoding on the q-ary symmetric channel. We construct several codes and provide simulation results to assess their performance. We end with conclusions and discussion in Section IV.

II. Linear form encoding and decoding

A. Linear form encoding

We denote the finite field with q elements by $\mathbb{F}_q$. Often $q = 2^m$ and the corresponding field is denoted by $\mathbb{F}_{2^m}$. Let C be a binary code of dimension K and length N. Furthermore, let G denote a generator matrix of C. Let,

$$s = (s_1, \ldots, s_K), \tag{1}$$

denote a vector of source symbols where $s_i \in \mathbb{F}_q$ for $i = 1, \ldots, K$. We assume that $q = 2^m$ and we can write each s_i as

$$s_i = (s_{im}, \ldots, s_{i1})^\top, \tag{2}$$

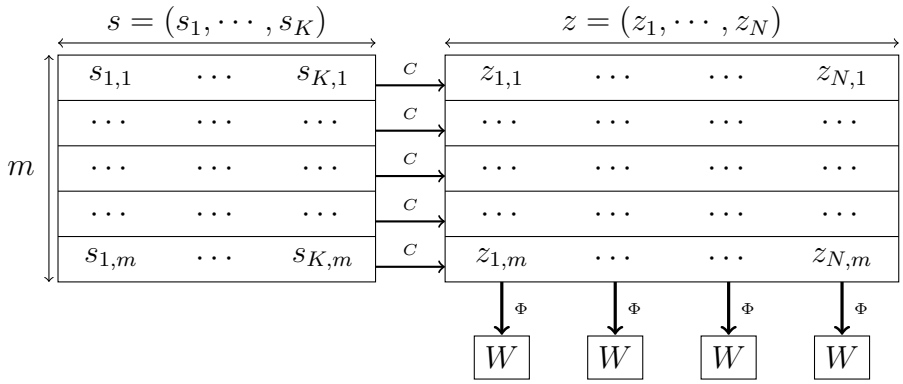

Fig. 1. Schematic diagram of linear form encoding and transmission on the channel W.

where $s_{ij} \in \mathbb{F}_2$. We view the source vector s as an $m \times K$ matrix where each column corresponds to the binary expansion of the corresponding element of $\mathbb{F}_q$. We encode s to obtain a codeword $z \in \mathbb{F}_q^N$ by applying the generator matrix of C to s

$$z = sG. \tag{3}$$

Each column of z we can identify with an element of $\mathbb{F}_q$ which shows that we can also view z as a vector of $\mathbb{F}_q^N$. Formally we have defined a code C_q as the tensor product of the binary code C as an $\mathbb{F}_2$-module with the $\mathbb{F}_2$-module $\mathbb{F}_q$.

$$C_q = C \otimes \mathbb{F}_q. \tag{4}$$

Now, consider a set Φ of n non-zero $\mathbb{F}_2$-linear forms of $\mathbb{F}_q$:

$$\Phi = \{\phi_i | i = 1, \ldots, n\}. \tag{5}$$

We assume that $\dim\langle\Phi\rangle = m$ and define an error-correcting code C_Φ as

$$C_\Phi = \{(\phi_1(x), \ldots, \phi_n(x)) \,|x \in \mathbb{F}_q\}. \tag{6}$$

This is a linear code of length n. Since we have assumed that $\dim\langle\Phi\rangle = m$, its dimension is m. Hence, C_Φ is an $[n, m]_2$-code. We will map the symbols of z to bit-vectors of length n using C_Φ. The main observation is that if we consider all the bits at a fixed position of the codewords from C_Φ, this sequence of bits is an element of C. To be more specific, let C_{ϕ_i} for a linear form ϕ_i be a code defined by

$$C_{\phi_i} = \{(\phi_i(z_1), \ldots, \phi_i(z_N)) | (z_1, \ldots, z_N) \in C_q\} \tag{7}$$

The following lemma is an immediate consequence of the fact that the code is defined by binary linear forms.

Lemma 1: The code C_{ϕ_i} is a binary linear code of dimension K and length N. Furthermore, $C_{\phi_i} = C$.

Proof: It follows from the definition of C_{ϕ_i} that for each codeword $c = (c_1, \ldots, c_N) \in C_{\phi_i}$ there exists a codeword $z = (z_1, \ldots, z_N) \in C_q$ with $\phi_i(z_j) = c_j$ for $j = 1, \ldots, K$ and vice versa. Each z corresponds to a vector of source symbols $s = (s_1, \ldots, s_K) \in \mathbb{F}_q^K$ for which

$$\phi_i \left(\sum_{l=1}^{K} s_l g_{lj} \right) = c_j, \tag{8}$$

where g_{lj} is the element of G at row l and column j. Furthermore, s_l is a column vector of bits corresponding to the binary expansion of the source symbol from $\mathbb{F}_q$. Equivalently, we can write (8) as

$$\phi_i \left(\sum_{l=1}^{K} s_l g_{lj} \right) = \sum_{l=1}^{K} \phi\left(s_l\right) g_{lj} = c_j,$$

and we conclude that for each $c \in C_{\phi_i}$ there exists a sequence of source symbols $s \in \mathbb{F}_q^K$ for which

$$(c_1, \ldots, c_N) = (\phi_i(s_1), \ldots, \phi_i(s_K))G, \tag{9}$$

and vice versa. Since $\{\phi_i(s_j) | s_j \in \mathbb{F}_q\} = \mathbb{F}_2$ and G is a generator matrix of C, the claim follows. ■

B. Linear form decoding

We consider transmission of a codeword $z \in C_q$ on a channel W

$$W : \mathbb{X} \mapsto \mathbb{Y}, \tag{10}$$

where the channel input alphabet $\mathbb{X}$ is taken to be $\mathbb{F}_q$ and $\mathbb{Y}$ is the channel output alphabet. The vector of channel outputs is denoted by $y = (y_1, \ldots, y_N)$. The corresponding random variables we denote by Z and Y, respectively. The channel is defined by a channel transition density which for simplicity we denote by $p(y_j|z_j)$. Furthermore, the channels we consider in this paper are all memoryless.

Given the channel output y we could decode C_q directly. However, in general optimal decoding of a code is only feasible for small N. Furthermore, for coding schemes for which low-complexity

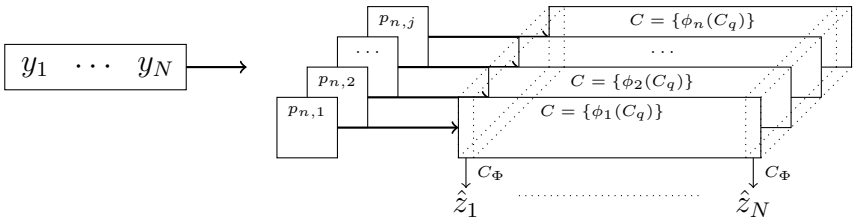

Fig. 2. Schematic diagram of linear form decoding.

q-ary decoding algorithms exist and perform well, their binary counterparts are usually far more attractive from a complexity point of view.

Note that we will not transmit the elements of the binary code C_Φ directly: the codewords we transmit over the channel are elements of $\mathbb{F}_q^N$, i.e., bit-vectors of length Nm. The idea is very similar to that of concatenated codes. However, with a concatenated code the length of the bit vector we transmit would be Nn and we would lose rate. In our case we are not losing rate at all. Figure 1 gives an overview of the encoding method.

Our approach to the decoding problem is as follows. First, we decode each of the binary codes C_{ϕ_i} for a properly chosen set of linear forms Φ. Since each of the C_{ϕ_i} is equal to C by Lemma 1 we can use a decoding algorithm suitable for C. In this paper we only explore the use of sparse graph codes for C and decode C by a BP algorithm. Once we have decoded each of the C_{ϕ_i}, we can use these results as an estimate for the value of each of the linear forms evaluated at the coordinate positions of the codeword z. For each coordinate position we decode C_Φ to refine the estimate of this position. As with C we can choose C_Φ depending on the requirements and the constraints on e.g. computational complexity.

C. Decoding C_{ϕ_i}

In general optimal decoding is prohibitively complex for most error-correcting codes. Instead of optimal decoding we consider codes which can be decoded by suboptimal low-complexity algorithms. In this paper the codes we consider for C are sparse graph codes and BP algorithms provide an efficient way to decode these codes.

Let $z = (z_1, \ldots, z_N)$ denote a codeword of C_q and consider the case that we are decoding linear form ϕ_i. The corresponding codeword from C_{ϕ_i} is denoted by $x = (x_1, \ldots, x_N)$. For each

of the coordinate positions of x we compute $\Pr[X_j = x_j | Y_j = y_j]$. We will denote this quantity by $p_{ij}(x_j)$ and we can compute it as

$$\begin{aligned} p_{ij}(x_j) &= \Pr[\phi_i(Z_j) = x_j | Y_j = y_j] \\ &= \sum_{\substack{a \in \mathbb{F}_q \\ \phi_i(a) = x_j}} \Pr[Z_j = a | Y_j = y_j]. \end{aligned} \tag{11}$$

We can express $\Pr[Z_j = a | Y_j = y_j]$ in terms of the channel transition density as follows

$$\begin{aligned} \Pr[Z_j = a | Y_j = y_j] &= \frac{\Pr[Y_j = y_j | Z_j = a] \cdot \Pr[Z_j = a]}{\Pr[Y_j = y_j]} \\ &= \frac{1}{\gamma_1} \Pr[Y_j = y_j | Z_j = a] = \frac{1}{\gamma_1} p(y_j | a), \end{aligned} \tag{12}$$

where $\gamma_1 = \Pr[Y_j = y_j] / \Pr[Z_j = a]$ is a normalization constant. We can interpret $p_{ij}(x_j)$ as follows. Once we have observed a channel output y_j, it induces an a posteriori probability distribution on the channel input alphabet $\mathbb{F}_q$. To compute the probability that a linear form ϕ_i evaluated on this alphabet is 0 or 1, we marginalize over this probability distribution. The marginal probability that linear form ϕ_i is equal to x_j is given by $p_{ij}(x_j)$.

In this paper we limit ourselves to one family of codes for C to illustrate the capabilities of linear form decoding. We use Fountain codes which are a type of rateless codes. A first practical family of Fountain codes are LT codes [12]. A second family of Fountain codes which build upon LT codes are Raptor codes as introduced in [13]. Raptor codes offer a better performance-complexity tradeoff than LT codes. The numerical results presented in Section III are obtained with LT-codes. Hence we give a summary of the important properties of LT-codes and how they fit within the linear form decoding framework. For more details about these codes we refer the reader to [12], [13], [14].

A binary LT code is defined by parameters $(K, \Omega(x))$ where K is the number of input bits of the LT code and $\Omega(x)$ the output distribution. The number of output symbols is given by N and an important property of an LT code is that N is not fixed a priori. For our purpose we will generate a single LT code and use it as the code for C as described in earlier sections. An LT code can efficiently be decoded by a BP algorithm as described in [14] and BP decoding is used to decode C_{ϕ_i}.

D. Decoding C_Φ

We assume that the decoding algorithm for C provides us with an estimate $\Pr\left[\phi_i(z_j) = c_i\right]$ for each of the ϕ_i evaluated at the coordinate position z_j of the codeword of C_q. We will assume that these estimates are independent of each other. In general this will not be the case, but our practical experiments suggest that correlations are only weak.

The purpose of decoding C_Φ is to refine the estimates of the coordinates of z. Let $c = (c_1, \ldots, c_n) \in C_\Phi$ and define a distribution on the codewords of C_Φ as

$$p_\Phi(c) = \frac{1}{\gamma_2} \prod_{i=1}^{n} \Pr\left[\phi_i(z_j) = c_i\right], \tag{13}$$

where γ_2 is a normalization constant chosen in such a way that

$$\sum_{c \in C_\Phi} p_\Phi(c) = 1. \tag{14}$$

Now, we can estimate a codeword from C_Φ as

$$\hat{c} = \underset{c \in C_\Phi}{\text{argmax}}\ p_\Phi(c). \tag{15}$$

In case the estimates of $\Pr[\phi_i(x) = 0]$ returned by the decoders of C are independent of each other, (15) is the optimal wordwise MAP decoder for C_Φ.

Once we have computed $p_\Phi(c)$, we can update for each ϕ_i the estimate of $\Pr[\phi_i(z_j) = c_i]$ as

$$\Pr[\phi_i(z_j) = c_i] = \sum_{\substack{x \in C_\Phi \\ x_i = c_i}} p_\Phi(x). \tag{16}$$

In this paper we will consider several options for Φ. For example we will choose Φ in such a way that C_Φ is a Hadamard code or a Hamming code or is equal to other suitable choices.

E. Linear form decoding algorithms

With the preparations of the previous sections, we are ready to describe our two decoding algorithms.

Algorithm 1: The set $\Phi = \{\phi_1, \ldots, \phi_n\}$ of linear forms is fixed throughout the algorithm. The algorithm has two phases: the C decoding phase phase and the C_Φ decoding phase.

(C) For each linear form $\phi_i \in \Phi$ we compute $p_{ij}(x_j)$ according to (11). The code C is decoded and the result is an updated estimate for $p_{ij}(x_j)$ which we denote by $\mu_{ij}(x_j)$

(C_Φ) For each coordinate position j we use $\mu_{ij}(x_j)$ to compute a distribution on the codewords of C_Φ according to (13) where the corresponding $\Pr[\phi_i(z_j)]$ is given by $\mu_{ij}(x_j)$. After decoding C_Φ, the final estimate of the jth symbol for the codeword from C_q is the inverse image of the codeword which has been found while decoding C_Φ.

Algorithm 1 can be extended in the following manner. The decoding algorithm for C may permit a partial decoding of C. For an LT code which is decoded by a BP algorithm we can perform several iterations of BP decoding and then decode C_Φ. Now the idea is to alternate between the two decoding phases. A relevant question is the number of iterations performed in decoding C before turning to decoding C_Φ. For this purpose we define a vector I which we call the iteration profile. The size of I designates the number of times we turn to decoding C_Φ and the jth element of I is the number of iterations we perform before decoding C_Φ the jth time. As we will see later the final performance depends on this choice. We denote this version of our algorithm by *Algorithm 2*.

Algorithm 2: The set $\Phi = \{\phi_1, \ldots, \phi_n\}$ of linear forms is fixed throughout the algorithm. The algorithm has two phases: the C decoding phase and the C_Φ decoding phase. We alternate between the C and C_Φ phases where at each C decoding phase we perform a number of iterations as defined by the iteration profile I.

(C) For each linear form $\phi_i \in \Phi$ we compute $p_{ij}(x_j)$ according to (11). The code C is decoded and the result is an updated estimate for $p_{ij}(x_j)$ which we denote by $\mu_{ij}(x_j)$

(C_Φ) For each coordinate position j we use the $\mu_{ij}(x_j)$ to compute a distribution on the codewords of C_Φ according to (13) where the corresponding $\Pr[\phi_i(z_j)]$ is given by μ_{ij}.

F. Complexity

The running time of these algorithms can be easily assessed. It's easy to show that *Algorithm 1* and *Algorithm 2* have a time and space complexity of the same order. In what follows, we derive a complexity analysis for *Algorithm 1*.

Since at each iteration of the BP algorithm n values are updated, the running time will be proportional to the number of edges in the decoding graph of the LT code times n. The running time of the C_Φ-part of the decoding process depends on the algorithm used. The straightforward implementation of $\Pr[x = c]$ for all $c \in C_\Phi$ uses $\leq n2^m$ operations on real numbers: for every vector c we need to calculate a product of at most n terms. (A more accurate estimate of the

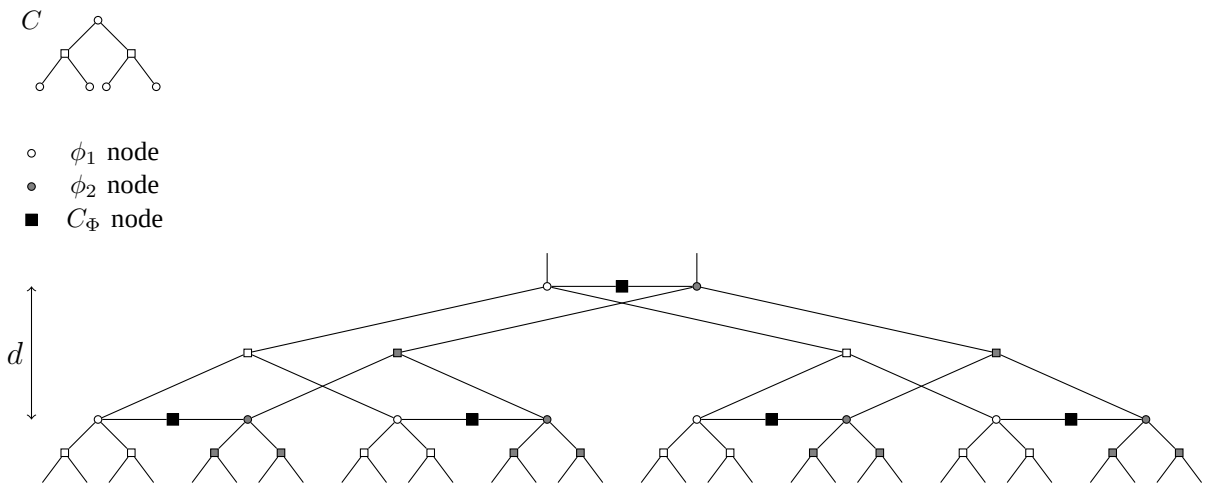

Fig. 3. Example of a computation tree for linear form decoding with two linear forms.

number of operations is $\sum_{c \in C} \mathrm{wgt}(c)$, where $\mathrm{wgt}(c)$ is the Hamming weight of c; hence, if C_Φ contains the all-one codeword, the running time of this algorithm will be at most $n2^m/2$.)

In total, the running time of the decoder is $O(Nan + n2^m)$, where a is the average output degree of the LT code. For N large, the running time is dominated by $O(Nan)$.

Depending on how we choose C_Φ, the running time can be anywhere between $O(Nam)$ and $O(Na2^m)$. Even in the worst case of $O(Na2^m)$ the running time is by a factor of m better than the running time of the BP algorithm over $\mathbb{F}_q$ (which is $O(Nam2^m)$).

In contrast to the q-ary belief-propagation algorithm, the new algorithms offers a tradeoff between the running time and the decoding capability.

G. Graphical interpretation of linear form decoding

Several parameters have to be chosen for *Algorithm 1* and *Algorithm 2*. For *Algorithm 2* the iteration profile defines how many iterations are spent at decoding C before switching to decoding C_Φ and back. In this section we give a graphical representation of linear form decoding which gives us some clues on how to choose the iteration profile and which factors may influence decoding performance.

We assume that a sparse graph code is used for C and consider part of the computation tree. The upper-left corner of Figure 3 gives an example of a computation tree of height $d = 2$ which

corresponds to 2 decoding iterations for C. Next, we consider the case that Φ consists of two linear forms: $\Phi = \{\phi_1, \phi_2\}$. The lower part of Figure 3 shows the computation tree of 2 rounds of *Algorithm 2* where one round corresponds to decoding C and C_Φ. The nodes of the graph corresponding to ϕ_1 and ϕ_2 are drawn with a white and gray filling, respectively. The large black squares correspond to the constraints induced by C_Φ.

In Figure 3, C is decoded for $d = 2$ iterations and then C_Φ is decoded. At this point information between the two graphs corresponding to ϕ_1 and ϕ_2 is exchanged. We observe that the constraints corresponding to C_Φ introduce cycles in the graph. This happens even when the graph corresponding to C is a tree as is the case in Figure 3. The length of these cycles is easily shown to be $4 + 2d$.

It is well-known that the presence of cycles deteriorates the performance of BP-based decoding algorithms. Hence it is of importance that we do not choose the number of iterations spent in decoding C too small since it directly corresponds to the length of the cycles created. Experimental results confirm that especially for the first few rounds d should not be chosen too small.

Note that in *Algorithm 1* we only perform decoding of C_Φ at the final step and there is no issue there. For *Algorithm 2* there is a trade-off between the gain the exchange of information between the graphs of linear forms can give us and the deterioration of performance due to the introduction of cycles.

III. Numerical results on the q-ary symmetric channel

In this section we describe some explicit constructions and give the results of numerical experiments. As mentioned in an earlier section we use an LT code for C_q and we perform experiments with several choices for C_Φ. We limit ourselves to the q-ary symmetric channel (q-SC) in this paper. This should not be seen as a limitation of our techniques since the q-SC shares the most important properties with more practical non-binary channels. For instance the PID capacity of the q-SC is less than its capacity. The restriction to the q-SC allows us to focus on linear form decoding itself.

The q-SC takes a q-ary symbol as its input and outputs either the unchanged input symbol, with probability $1 - p$, or any of the other $q - 1$ symbols, with probability $\frac{p}{q-1}$. The capacity of

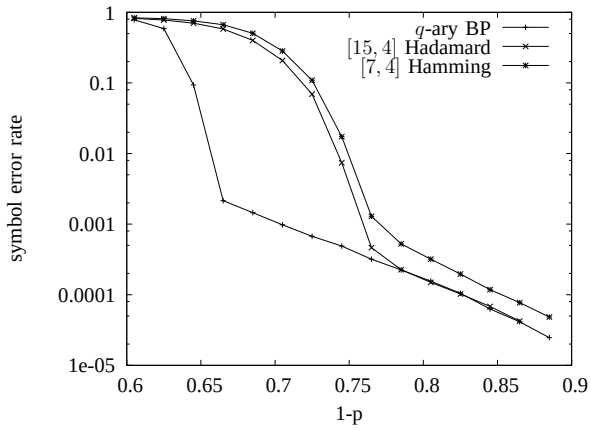

Fig. 4. Performance of *Algorithm 1* on the q-ary symmetric channel with $q = 16$ for $K = 6000$ and $N = 15000$.

the q-SC with crossover probability p is

$$\text{Cap}_{q\text{-SC}} = m - h(p) - p\log_2(q-1)$$

bits per channel use, where $h(p)$ denotes the binary entropy function.

In this section, we present simulations results of an LT-code of block length N over $\mathbb{F}_{2^m}$, whose parameters are $(k, \Omega(x))$. We also compare its performance to q-ary belief propagation. All experimental results were obtained using the output degree distribution

$$\begin{aligned}\Omega(x) = {} & 0.007690x + 0.493570x^2 + 0.166220x^3 + 0.072646x^4 \\ & + 0.082558x^5 + 0.056058x^8 + 0.037229x^9 + 0.055590x^{19} \\ & + 0.025023x^{65} + 0.003135x^{66}.\end{aligned}$$

1) Performance of Algorithm 1*:* Figure 4 compares the symbol-error rate (SER) of *Algorithm 1* of an LT code over the field $\mathrm{GF}(2^4)$. Furthermore, K and N are chosen as 6000 and 15000, respectively. The number of BP iterations is 80. For C_Φ the $[15, 4]$-Hadamard code and the $[7, 4]$-Hamming code are used. We observe that once the SER is sufficiently low, the performance of *Algorithm 1* with the $[15, 4]$-Hadamard code approaches that of full BP decoding. For the $[7, 4]$-Hamming code the performance is slightly worse. On the other hand *Algorithm 1* performs worse for high and moderate SERs. The threshold of q-ary BP decoding is significantly lower.

Figure 5 compares the symbol-error rate (SER) of *Algorithm 1* of an LT code over the field $\mathrm{GF}(2^8)$. For C_Φ the $[255, 8]$-Hadamard code, a $[29, 8]$-code and a $[17, 8]$-code are used. The

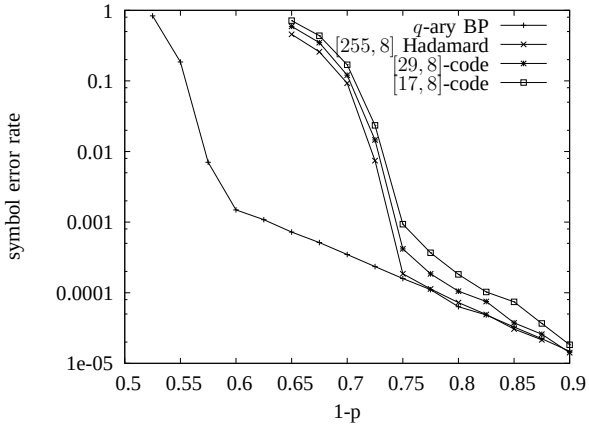

Fig. 5. Performance of *Algorithm 1* on the q-ary symmetric channel with $q = 256$ for $K = 6000$ and $N = 15000$.

$$G = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Fig. 6. The generator matrix of the $[17, 8]$ code.

$[17, 8]$-code is obtained by shortening a cyclic $[18, 9, 6]$-code. The generator matrix of this code is shown in Figure 6. The $[29, 8]$-code is obtained by shortening an extended $[32, 11, 12]$-BCH code on three positions. The generator matrix of this code is shown in Figure 7. Both these codes are optimal in the sense that they have the largest minimum distance given their length and dimension. The results for low SERs are similar as for the field $\mathrm{GF}(2^4)$. For low SERs the $[255, 8]$-Hadamard code has a SER comparable to q-ary BP decoding. The $[29, 8]$-code and the $[17, 8]$ perform slightly worse. However, the number of linear forms used for these two codes is considerably smaller (29 and 17, respectively). This gives a decoding complexity which is about ten times less compared to the Hadamard code. For low to moderate SERs *Algorithm 1*

$$G = \begin{bmatrix}
1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1
\end{bmatrix}$$

Fig. 7. The generator matrix of the $[29, 8]$ code.

Iteration profile	Iteration vector I
A	$[100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100]$
B	$[80, 80, 80, 80, 80, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10]$
C	$[20, 20, 20, 20, 20, 20, 20, 20, 20, 20]$

Fig. 8. The iteration profiles for *Algorithm 2*.

performs worse than the q-ary BP decoding algorithm as is the case for the field $\mathrm{GF}(2^4)$.

2) Performance of Algorithm 2*:* Figure 9 compares the SER performance of *Algorithm 2* with q-ary BP decoding of an LT code over the field $\mathrm{GF}(2^4)$. Furthermore, K and N are chosen as 6000 and 15000, respectively. As a reference, the performance of *Algorithm 1* is shown as well. The figure shows the performance of *Algorithm 2* for several iteration profiles. These profiles are labeled and defined in the table of Figure 8. In iteration profile A the total number of iterations is 1500 and each C decoding phase consists of 100 iterations. Compared to the other profiles the performance is best and a significant gain with respect to the performance of *Algorithm 1* is obtained. As expected the q-ary BP algorithm (80 iterations) performs best. However, *Algorithm 2-A* closes a large part of the gap between BP and *Algorithm 1*. Even though the total number of iterations of *Algorithm 2-A* is larger than the number of iterations we have used for q-ary BP decoding, the complexity of *Algorithm 2-A* is substantially less than q-ary BP decoding.

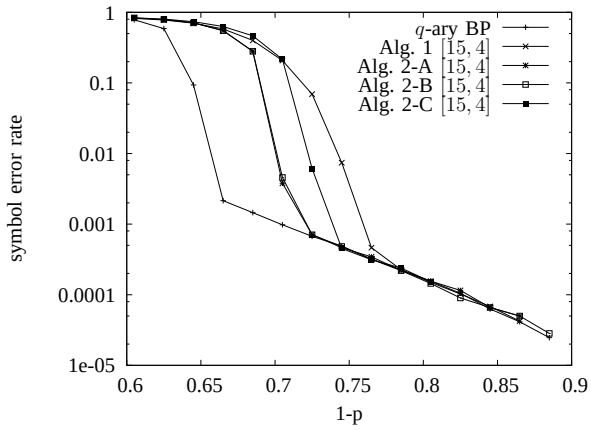

Fig. 9. Performance of *Algorithm 2* on the q-ary symmetric channel with $q = 16$ for $K = 6000$ and $N = 15000$.

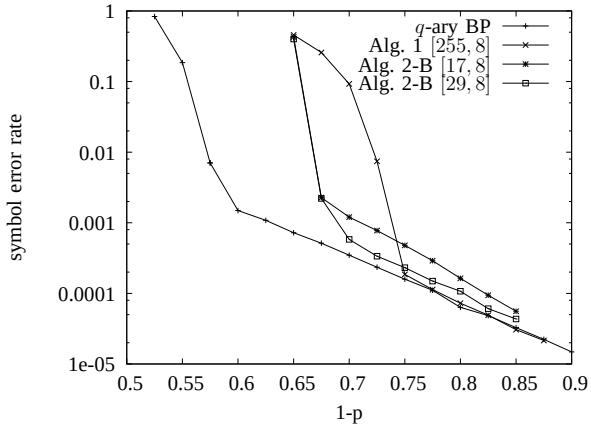

Fig. 10. Performance of *Algorithm 2-B* on the q-ary symmetric channel with $q = 256$ for $K = 6000$ and $N = 15000$.

Algorithm 2-B and *Algorithm 2-C* use a total number of iterations of 550 and 200, respectively. We observe that *Algorithm 2-B* performs almost the same as *Algorithm 2-A*. *Algorithm 2-C* performs a little bit worse than *Algorithm 2-A*. However, the decoding complexity of *Algorithm 2-C* is less compared to the other two iteration profiles.

Figure 10 compares the SER performance of *Algorithm 2* with q-ary BP decoding of an LT code over the field $\mathrm{GF}(2^8)$ where $K = 6000$ and $N = 15000$. As a reference, the performance of *Algorithm 1* is shown as well. The figure shows the performance of *Algorithm 2* for iteration profile B where C_Φ is chosen as the $[17, 8]$-code and the $[29, 8]$ code. As with the field $\mathrm{GF}(2^4)$, *Algorithm 2* closes part of the gap between *Algorithm 1* and BP. With respect to threshold of

the code, the $[17, 8]$-code and the $[29, 8]$ code behave the same. The $[29, 8]$ code has a better performance for low SERs.

IV. Conclusion

We provided a class of algorithms for decoding binary codes on q-ary channels. In contrast to the well-known q-ary belief-propagation algorithm, the new algorithm provides a tradeoff between the complexity of the decoding and the decoding capability of the algorithm. Several simulation results have been provided to show the performance of the new algorithms as compared to the q-ary belief-propagation algorithm.

It is important to note that our algorithms cannot be used when the underlying code is not binary (i.e., if it is not the tensor product of a binary code with $\mathbb{F}_q$). On the other hand, one cannot expect binary codes to perform as well as q-ary ones, so that the question arises to what extent the algorithms of this paper are interesting for practical code design. One answer to this question is that our algorithms are expected to perform well on non-standard channels, for example channels that arise in compression algorithms. In fact, these algorithms were invented for faster decoding of codes specifically arising in these applications. The algorithm is applicable to any binary linear code, not just LT codes. In particular, if we have a binary code with an efficient binary decoding algorithm, we can use the same code and the methods of this paper to decode the code when used with q-ary symbols rather than binary ones.

References

[1] H. Imai and S. Hirakawa, "A new multilevel coding method using error-correcting codes," *IEEE Trans. Inform. Theory*, vol. 23, no. 3, pp. 371–377, 1977.

[2] U. Wachsmann, F. H. Fischer, and J. B. Huber, "Multilevel codes: Theoretical concepts and practical design rules," *IEEE Trans. Inform. Theory*, vol. 45, no. 5, pp. 1361–1391, 1999.

[3] G. Caire, G. Taricco, and E. Biglieri, "Bit-interleaved coded modulation," *IEEE Trans. Inform. Theory*, vol. 44, pp. 927 – 946, May 1998.

[4] M. Davey and D. MacKay, "Low Density parity Check Codes over GF(q)," *IEEE Commun. Lett.*, vol. 2, pp. 165–167, June 1998.

[5] D. Mackay and M. Davey, "Evaluation of Gallager codes for short block length and high rate applications," in *In Codes, Systems and Graphical Models*, pp. 113–130, Springer-Verlag, 2000.

[6] T. Richardson and R. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Transactions on Information Theory*, vol. 47, pp. 599–618, 2001.

[7] D. Declerq and M. Fossorier, "Decoding algorithms for nonbinary LDPC codes over GF(q)," *IEEE Transactions on Communications*, vol. 55, pp. 633–643, 2007.

[8] M. Luby and M. Mitzenmacher, "Verification codes," in *Proceedings of the 40th Annual Allerton Conference on Communication, Control, and Computing*, 2002.

[9] A. Shokrollahi and W. Wang, "LDPC codes with rates very close to the capacity of the q-ary symmetric channel for large q," in *Proceedings of the International Symposium on Information Theory, Chicago*, 2004.

[10] R. Karp, M. Luby, and A. Shokrollahi, "Verification decoding of Raptor codes," in *Proc. of ISIT 2005*, pp. 1310–1314, 2005.

[11] H. Cronie, B. N. Ndzana, and A. Shokrollahi, "Decoding algorithms for binary raptor codes over nonbinary channels," in *Proc. of the IEEE Int. Symposium on Inform. Theory*, pp. 2487–2491, 2009.

[12] M. Luby, "LT-codes," in *Proc. of STOC*, pp. 271–280, 2002.

[13] A. Shokrollahi, "Raptor codes," *IEEE Trans. Inform. Theory*, vol. 52, pp. 2551–2567, June 2006.

[14] O. Etesami and M. A. Shokrollahi, "Raptor codes on binary memoryless symmetric channels," *IEEE Transactions on Information Theory*, vol. 52, pp. 2033–2051, 2006.