ABSTRACT

In this paper, we propose the tree-structured expectation propagation (TEP) algorithm for low-density parity-check (LDPC) decoding over the additive white Gaussian noise (AWGN) channel. By imposing a tree-like approximation over the graphical model of the code, this algorithm introduces pairwise marginal constraints over pairs of variables, which provide joint information of the variables related. Thanks to this, the proposed TEP decoder improves the performance of the standard belief propagation (BP) solution. An efficient way of constructing the tree-like structure is also described. The simulation results illustrate the TEP decoder gain in the finite-length regime, compared to the standard BP solution. For code lengths shorter than $n = 512$, the gain in the waterfall region achieves up to 0.25 dB. We also notice a remarkable reduction of the error floor.

1. INTRODUCTION

Nowadays, there is an increasing interest in inference techniques in the field of communications, especially in coding theory. The best example is the rebirth of low-density parity-check (LDPC) codes, which came with the popularization of belief propagation (BP) as an efficient message-passing algorithm [1, 2]. Since then, LDPC codes are extensively used due to their great performance [2, 3], and belief propagation (BP) is the standard technique for decoding [4]. This inference algorithm provides the maximum a posteriori (MAP) solution in tree-like graphs [5]. But it does not provide optimal estimations in graphs with cycles, or even it is not able to converge at all [5]. Since practical code lengths involves the existence of cycles in the graph, BP performance is restrained in practical scenarios.

This work was partially funded by Spanish government (Ministerio de Educación y Ciencia, TEC2009-14504-C02-01,02), Consolider-Ingenio 2010 CSD2008-00010), Universidad Carlos III de Madrid/Comunidad de Madrid (CCG10-UC3M/TIC-5304) and European Union (FEDER).

Expectation propagation (EP) algorithm [6] is an approximate inference tool that can be understood as a generalization of BP in two ways: it allows to consider continuous random variables and can be formulated to include more flexible approximating factorizations [7]. In a novel work, we proposed the TEP decoder for LDPC codes over the BEC [8], borrowing from the tree-structured EP algorithm [7], an extension of EP. Whereas the BP decoder for BEC iterates through the Tanner graph by looking for degree-one check nodes, the TEP decoder also looks for relations between variables through degree-two check nodes, which might reveal erased variables. The asymptotic behavior and scaling laws for the TEP decoder have been also studied in [9]. The generalized TEP (GTEP) decoder, proposed in [10], extends the idea of the TEP decoder by considering relations between any number of variables. Since the GTEP decoder provides the MAP solution over the BEC, it is jointly presented with regular LDPC codes as a capacity achieving scheme [11].

Both TEP and GTEP decoders are comparable to existing techniques for efficient Gaussian elimination [12, 13]. However, since all these methods perform Gaussian elimination over the parity-check matrix $H$, they can only be applied when transmitting over erasure channels. Nevertheless, the formulation of the TEP decoder, the tree-structured EP [7], is also extensible to more general binary symmetric memoryless channels.

In this paper, we propose the TEP decoder for LDPC decoding over the additive white Gaussian noise (AWGN) channel. By imposing in the factor graph a tree-like structure of relations between pairs of variables, the TEP decoder also calculates and propagates pairwise probabilities, capturing the interactions between these variables. Hence, while the computed marginal probabilities of two uncertain (with an a priori probability close to 0.5) variables connected to the same check node remains ambiguous, its joint pairwise probability may capture how likely both variables are. This information is stored in the conditional factor, and it is propagated...
through the graph, improving the decoding. We show that the TEP decoder outperforms the BP algorithm in the finite-length regime. We also propose an intuitive method that, even for the sparse matrix of an LDPC code, is able to find strongly dependent pairs of variables that allow the TEP decoder to outperform BP. However, the most efficient way of joining the variables in the graph remains an open problem.

The paper is organized as follows. In Section 2 we describe the formulation of the tree-structured EP algorithm, and provides a first insight into its benefits. The method considered for choosing and constructing the tree-like approximation is presented in Section 3. Experimental results in Section 4 help to illustrate the benefits of the TEP decoder. Finally, Section 5 ends with conclusions and some proposals for future work.

2. TREE-STRUCTURED EXPECTATION PROPAGATION ALGORITHM

Consider an LDPC binary code $C$ with parity check matrix $H$, of size $k \times n$, where $k = n(1 - r)$, $n$ is the code length and $r$ the rate of the code. Any codeword $v$ in $\mathbb{F}_2^n$ belongs to the code $C$ as long as $v^T H = 0$. Each row of $H$ imposes a zero parity constraint between a subset of variables:

$$
\mathcal{C}_j(v) = \mathbb{I}\left[\sum_{i \in I_j} v_i \mod 2 = 0\right] \quad \forall j = 1, \ldots, k,
$$

(1)

where $v_i$ is the $i$-th component of $v$, and $I_j$ is the set of non-zero entries in the $j$-th row of $H$. The indicator function $\mathbb{I}[:]$ takes value one when the boolean relation is satisfied.

Assuming $y$ as the observed channel output, a bit-MAP decoder [14] estimates the transmitted vector $\hat{v} = [\hat{v}_1, \hat{v}_2, \ldots, \hat{v}_n]$ as follows:

$$
\hat{v}_u = \arg\max_{v \in \{0,1\}} \sum_{v \in C:v_u=v} p(v|y)
= \arg\max_{v \in \{0,1\}} \sum_{v \in C:v_u=v} p(y|v)p(v)
= \arg\max_{v \in \{0,1\}} \prod_{v:v_u=v} p(y_i|v_i) \prod_{j=1}^k \mathcal{C}_j(v)
$$

(2)

for $u = 1, \ldots, n$, a memoryless channel and equally probable codewords $v$.

The factor graphs of most of the LDPC codes of interest present cycles, and thereby a decoding complexity that grows exponentially with $n$ [5]. BP [2] is the standard algorithm to approximately solve this problem in linear time. BP can be cast as an approximation of $p(v|y)$ in (2) by a complete disconnected factor graph, i.e.

$$
p(v|y) \propto \prod_{i=1}^n p(y_i|v_i) \prod_{j=1}^k \mathcal{C}_j(v) \approx \prod_{i=1}^n \hat{q}_{i,BP}(v_i),
$$

(3)

where $\hat{q}_{i,BP}(v_i)$ is the estimate for the $i$-th variable [5].

2.1. Tree-EP algorithm for LDPC decoding

The Tree-EP algorithm [7] improves BP decoding because it approximates the posterior $p(v|y)$ in (2) as $\hat{q}(v)$, which presents a tree (or forest) Markov-structure between sets of pairs of variables $\{(i, p_i)\}_{i=1}^n$, i.e.:\n
$$
\hat{q}(v) = \prod_{i=1}^n \hat{q}_i(v_i|v_{p_i}),
$$

(4)

where $\hat{q}_i(v_i|v_{p_i})$ approximates the conditional probability $p(v_i|v_{p_i}, y)$. Note that if $p_i$ is empty, such term is approximate through a single-variable factor $\hat{q}_i(v_i)$. The optimal choice of $\hat{q}(v)$ is such that it minimizes the inclusive Kullback-Leibler (KL) divergence with $p(v|y)$:

$$
\hat{q}(v) = \arg\min_{q(v) \in \mathcal{F}_{\text{tree}}} D_{KL}(p(v|y) || q(v)),
$$

(5)

where $\mathcal{F}_{\text{tree}}$ is the family of all discrete probability distributions that factorize according to (4).

Compared to the fully disconnected approximation for the BP in (3), the additional relations introduced by the TEP algorithm in the factor graph provide more accurate estimates for the marginals $p(v_u|y)$. These joint factors capture information about the joint p.d.f. of pairs of variables that is then propagated through the graph. When no relations between variables are considered, i.e. $p_i = \emptyset \forall i$, we lead to the fully disconnected approximation of the BP, and thus the EP converges to the BP solution [15].

The computation of the marginals in (4) has the same complexity as (2). The Tree-EP algorithm overcomes this problem by introducing the following approximation for the parity-check constraints:

$$
W_j(v) = \prod_{z=1}^n w_{z,j}(v_z, v_{p_i}),
$$

(6)

where $w_{z,j}(v_z, v_{p_i})$ are non-negative real functions. The TEP performs updating iteratively $\hat{q}(v)$. Let denote with $q^\infty(v)$ the Tree-EP solution after convergence, and with $\ell$ the update of each term at the end of iteration $\ell$, i.e. $W_j(v)$ and $q^\ell(v)$. The Tree-EP approximation to the posterior $p(v|y)$ at each
iteration yields to:

\[
q^{f}(v) = \prod_{z=1}^{n} q^{f}_{z}(v_{z} | v_{p_{z}}) = \prod_{z=1}^{n} p(y_{z} | v_{z}) \prod_{j=1}^{k} W^{f}_{j}(v) = \prod_{z=1}^{n} \left( p(y_{z} | v_{z}) \prod_{j=1}^{k} w^{f}_{j,z}(v_{z}, v_{p_{z}}) \right). \tag{7}
\]

In Algorithm 1 is described in detail how the TEP decoder performs. At iteration \(\ell\), only the \(m\)-th factor, \(W^{f}_{m}(v)\), is refined. In Step 7 we replace \(W^{f-1}_{m}(v)\) by the true value \(C_{m}(v)\) in the \(q^{f-1}(v)\) function. The resulting function is denoted by \(f(v, \ell, m)\). Then, the approximation to \(q^{f}(v)\) is obtained as the minimization of the KL divergence \([16, 17]\), which is done by moment matching, as depicted in Steps 8 to 10:

\[
q^{f}(v) = \arg \min_{q(v) \in F_{tree}} D_{KL} \left( f(v, \ell, m) \parallel q(v) \right). \tag{8}
\]

The computation of the marginals \(q(v_{i} | v_{p_{i}})\) for \(i = 1, \ldots, n\) over \(f(v, \ell, m)\) in (13) and (14) can be performed efficiently. Let us first express the factorization of \(f(v, \ell, m)\) in a more convenient way:

\[
f(v, \ell, m) = C_{m}(v) \frac{q^{f-1}(v)}{W^{f-1}_{m}(v)} = C_{m}(v) \prod_{z=1}^{n} p(y_{z} | v_{z}) \prod_{j=1}^{k} W^{f-1}_{j}(v) = C_{m}(v) \prod_{z=1}^{n} g^{f-1,m}_{z}(v_{z}, v_{p_{z}}), \tag{9}
\]

where we have introduced the following auxiliary functions

\[
g^{f-1,m}_{z}(v_{z}, v_{p_{z}}) = p(y_{z} | v_{z}) \prod_{j=1}^{k} w^{f-1}_{z,j}(v_{z}, v_{p_{z}}). \tag{10}
\]

Therefore, the marginalization of (9) in (13) yields

\[
q(v_{i}, v_{p_{i}}) = \sum_{v \sim \{v_{i}, v_{p_{i}}\}} C_{m}(v) \prod_{z=1}^{n} g^{f-1,m}_{z}(v_{z}, v_{p_{z}}), \tag{11}
\]

where terms \(g^{f-1,m}_{z}(v_{z}, v_{p_{z}}), z = 1, \ldots, n\) maps over the same factor graph than the tree structure chosen in (4).

The presence of cycles in the factor graph of \(f(v, \ell, m)\) is due to the parity factor \(C_{m}(v)\). The graph is cycle-free as long as, among the variables connected to the parity check node \(C_{m}(v)\), none of them are linked by a conditional term \(q(v_{i} | v_{p_{i}})\) in (4), as illustrated in Fig. 1(a). Otherwise the graph presents cycles, as shown in Fig. 1(b). In the first case, the marginal computation in (13) is solved at linear cost by message passing. For the latter, where the graph is not completely cycle-free, we can compute the pairwise marginals using Pearl’s cutset conditioning algorithm \([18]\). By assuming as known some of the variables involved, this algorithm breaks the cycles and allows an efficient estimation of the marginal and pairwise terms by message-passing. These cycles, and the way they are processed through pairwise terms, play a crucial role in understanding why the TEP algorithm outperforms the BP solution.

**Algorithm 1** Tree-EP algorithm for a predefined tree structure.

1: \(\ell = 0\).
2: Initialize \(W^{f}_{j}(v) = 1\) for \(j = 1, \ldots, k\).
3: Initialize \(q^{0}(v) = \prod_{i=1}^{n} p(y_{i} | v_{i})\).
4: repeat
5: \(\ell = \ell + 1\).
6: Chose a \(W^{f}_{m}(v)\) to refine: \(m = \text{mod}(\ell, k)\).
7: Remove \(W^{f-1}_{m}(v)\) from \(q^{f-1}(v)\) and include the true term \(C_{m}(v)\):

\[
f(v, \ell, m) \equiv C_{m}(v) \frac{q^{f-1}(v)}{W^{f-1}_{m}(v)}.	ag{12}
\]

8: Compute \(q^{f}_{i}(v_{i}, v_{p_{i}})\) and \(q^{f}_{i}(v_{i})\) for \(i = 1, \ldots, n:\)

\[
q^{f}_{i}(v_{i}, v_{p_{i}}) \propto \sum_{v \sim \{v_{i}, v_{p_{i}}\}} f(v, \ell, m), \tag{13}
\]

\[
q^{f}_{i}(v_{i}) = \sum_{v_{p_{i}}} q^{f}_{i}(v_{i}, v_{p_{i}}). \tag{14}
\]

9: Compute \(q^{f}_{i}(v_{i} | v_{p_{i}})\) for \(i = 1, \ldots, n:\)

\[
q^{f}_{i}(v_{i} | v_{p_{i}}) = \frac{q^{f}_{i}(v_{i}, v_{p_{i}})}{q^{f}_{i}(v_{p_{i}})} \quad \text{for} \quad i = 1, \ldots, n.
\]

10: \(q^{f}(v) = \prod_{i=1}^{n} q^{f}_{i}(v_{i} | v_{p_{i}})\).
11: Compute \(w^{f}_{i,m}(v_{i}, v_{p_{i}})\) from \(q^{f}(v)\). By (7),

\[
w^{f}_{i,m}(v_{i}, v_{p_{i}}) = \frac{q^{f}_{i}(v_{i} | v_{p_{i}})}{p(y_{i} | v_{i}) \prod_{j=1}^{k} w^{f-1}_{i,j}(v_{i}, v_{p_{i}})} \quad \text{for} \quad i = 1, \ldots, n.
\]

12: \(W^{f}_{m}(v) = \prod_{i=1}^{n} w^{f}_{i,m}(v_{i}, v_{p_{i}})\).
13: until all \(W^{f}_{j}(v), j = 1, \ldots, k\), converge.
14: \(q(v) \approx q^{\text{opt}}(v)\).
Fig. 1. Example of factor graph associated to \( f(\ell, m) \). In (a), the graph is cycle free since \( \mathcal{C}_m \) does not connect any variable \( V_i \) with its parent \( V_{p_i} \). In (b), the graph has a cycle since both \( V_r \) and \( V_{p_r} = V_o \) belong to \( I_m \).

3. TREE STRUCTURE CONSTRUCTION

As described in the previous section, the TEP decoder imposes a tree structure over the factor graph of the code. Thus, which variables are selected to be joined is a key factor in the performance of the TEP decoder. A straightforward solution would be to join all the variables by pairwise factors. However, most of these joint factors will not provide any useful information, i.e. the variables related are already revealed, and the joint probability does not provide any additional information over the marginals. Besides, the additional complexity would not be justified, and this solution is discarded. Therefore, in this section we approach this problem and provide a simple, yet meaningful, technique based on the analysis of the TEP decoder for the BEC.

As pointed out by Minka in [7], it is more useful to join variables with higher mutual information, capturing this way more information from the graph. This fact was previously observed in the analysis of the TEP decoder for the BEC [9], where we looked for degree-two nodes, i.e. parity checks with only two erased (probability of 0.5) variables. By joining them we capture if the variables are either equal or different. Equivalently, when transmitting over a binary AWGN channel, cycles in the graph only constituted by mostly ambiguous variables, i.e. with a priori probabilities far from 0 or 1, might also lead to equivalent scenarios, where the probability estimates could not converge. If the variables of the cycle are joined by a chain of pairwise factors, the TEP decoder is able to provide to and extract from the cycle the joint probability of the variables connected to the processed check node. Thus, all this information is propagated through the graph as the decoding process goes forward, improving the performance of the TEP decoder.

Previously to construct the tree structure, we need to find cycles in the graph. We propose to look for a cycle as follows. We refer to the set of uncertain variables, \( V^n \), as the one constituted by the variables who satisfies \( |p(y_i | v_i) - 0.5| \leq \delta \) for \( i = 1, \ldots, n \), and threshold \( 0 \leq \delta \leq 0.5 \). Sets \( u = \{u_1, \ldots, u_r\} \) and \( s = \{s_1, \ldots, s_r\} \) contain the indexes of the variables and checks found for the cycle, and \( C^m_r \) and \( V^n_r \) are the checks and variables connected to \( V_{u_r} \) and \( C_{s_r} \), respectively. Then we run the steps below,

- Initialization: set \( r = 1, u = s = \emptyset \).
- Initial node of the chain: from set \( V^n_u \), randomly pick a variable \( V_i \) as the initial node \( u_1 = i \). Set \( r = 2 \).
- Step 1: if \( C^m_{r-1} \subseteq C_{s_r} \), initialize the algorithm again. Otherwise, from set \( C^m_{r-1} \), we randomly choose a check \( C_j \) such that \( j \notin s \). Index \( j \) is included in \( s \) as \( s_r \).
- Step 2: if \( r > 2 \) and \( V_{u_1} \in V^n_{r-1} \), go to Step 4.
- Step 3: from \( V^n_{r-1} \) select the most uncertain variable \( V_i \), i.e. with channel likelihood closest to 0.5, such that \( i \notin u \), and include it in set \( u \) as \( u_r \). Set \( r = r + 1 \) and go to Step 1.
- Step 4: the algorithm successfully finishes and the cycle leads \( u = \{u_1, u_2, \ldots, u_r\} \).

The previous algorithm provides a cycle. This cycle defines a chain of variables, related by pairwise factors \( q(v_{u_z} | v_{u_{z-1}}) \), with \( z = 2, \ldots, r \). Thus, we can run the algorithm \( t \) times, from a different initial variable, to get \( t \) chains. Since we have to restrict the tree structure to disjoint chains, we have to select a subset of them, \( t' \), to form vector \( V^S \). Finally, the pairwise factors are introduced in the graph of the code according to the chains in \( V^S \).

The method considered to construct \( V^S \) is not unique. The algorithm proposed, which is an attempt to emulate how the TEP decoder over the BEC performs, works remarkably well in practice. Nevertheless, other approaches can be considered. The design of other methods, as well as the implications of the particular factorization of \( q(v) \), remain an open problem.
4. SIMULATION RESULTS

We illustrate the BP and TEP performances for the (3,6)-regular LDPC ensemble. In [11], it is shown that we can find codes in the ensemble that present stopping sets (SSs) not smaller than \( s_{\text{min}} \), where \( s_{\text{min}} = 6, 6, 12, 24 \) for \( n = 128, 256, 512, 1024 \). In this section we show the BP and TEP decoders performance for the (3,6)-regular LDPC ensemble, expurgated for these \( s_{\text{min}} \) values. In Fig. 2 and 3 we depict the results in terms of bit error rate (BER) and word error rate (WER), respectively. The curves are averaged over 100 random samples of the ensemble.

For this expurgated ensemble, the TEP decoder exhibits an improvement compared to the BP, both in the waterfall region and in the error floor. In the waterfall region it has a gain of 0.3 dB for \( n = 128 \) and \( \text{BER} = 10^{-4} \). We also observe that the gain decreases with the code length \( n \). However, it is still around 0.2 dB for \( n = 1024 \) and a \( \text{BER} = 10^{-4} \). Similar conclusions can be drawn from the analysis of the WER. This behavior was already observed in [19] for the BEC. Then, in [9], it was proved that as \( n \) tends to infinity the gain vanishes, as there is no improvement in threshold.

In addition, we can observe that the gain increases as we approach the error floor region, yielding to a reduction in the error floor compared to the BP. This gain suggests that there is a fraction of short SSs that are now broken thanks to the inference of pairwise relationships.

5. CONCLUSIONS AND FUTURE WORK

In previous works we have proved that the TEP decoder improves the BP for finite-length codes. However, our results have been limited to the BEC. The peeling formulation of the TEP for the BEC facilitates the selection of the most useful chains, as they are constructed on the fly. However, in other channels no variables can be revealed until the end, and the selection of the chain of variables that form the tree structure is, by no means, straightforward.

In this paper, we successfully apply the tree-structured expectation propagation to the AWGN channel. We provide the formulation of the tree-structured EP algorithm to later propose how to construct the tree (or forest) Markov-structure that the TEP decoder imposes over the code graph. The simulation results illustrate the gain in performance of the TEP decoder, compared to the BP solution. As we can appreciate in the expurgated curves, and as for the BEC, we have larger gains for shorter codes.

The tree structure imposed by the TEP decoder is of major importance. The technique considered, based on the intuition derived from the analysis of the TEP decoder over the BEC, has shown to be quite meaningful. The design of other approaches to improve this method remain as further work.

6. REFERENCES


[10] Luis Salamanca, Pablo M. Olmos, Juan José Murillo-Fuentes, and Fernando Pérez-Cruz, “Tree expectation propagation for ML decoding of LDPC codes over the BEC,” *Submitted to IEEE Transactions on Communications*, 2011.


