An application of Generalized Belief Propagation: splitting trapping sets in LDPC codes

Jean-Christophe Sibel, Sylvain Reynal, David Declercq

An application of Generalized Belief Propagation: splitting trapping sets in LDPC codes

Jean-Christophe Sibel
UEB, IETR
INSA de Rennes, CNRS UMR 6164
Rennes, France
jean-christophe.sibel@insa-rennes.fr

Sylvain Reynal and David Declercq
ETIS, ENSEA
Univ. of Cergy-Pontoise, CNRS UMR 8051
Cergy-Pontoise, France
{reynal,declercq}@ensea.fr

***Abstract*—Generalized belief propagation (GBP) is known to be a well-suited technique for approximate inference problems in loopy factor graphs. It can absorb problematic subgraphs inside regions to reduce their influence on the inference. However, the choice of regions to be used in GBP remains a delicate issue. This paper proposes an approach to create specific regions when dealing with Low-Density Parity-Check (LDPC) codes. We split trapping sets, known to degrade the decoding performance, to make GBP locally optimal. Experiments show that GBP can then perform better than BP, especially in the error-floor region.**

***Index Terms*—Generalized Belief Propagation, clustering, LDPC codes, trapping set, error-floor.**

I. Introduction

Belief propagation (BP) is known as a reliable algorithm to approximate inference in factor graphs [1] in a wide variety of disciplines, including statistical physics, artificial intelligence, and digital communications. Originally created by Pearl [2], BP is likely to exhibit suboptimal performance when graphs have a loop-like topology [3], [4]. Alternative algorithms, searching for the same solutions as BP, have been developed but are relatively slow to converge [5], [6], [7].

A widespread approach for dealing with loopy factor graphs is to convert them to equivalent loopfree graphical models. For the most of practical cases, such an approach is intractable because the size of the systems is too large. Approximate methods are then necessary. The Region-Based Approximation (RBA) can do so absorbing loopy subgraphs in larger nodes, called regions inside a Bayesian network called the region-graph [8], [9]. The Generalized Belief propagation (GBP) is an iterative algorithm that passes messages between regions along the edges of the region-graph. Messages are then used to compute the marginal probabilities for regions, that solves the inference. GBP is able to dramatically outperform BP if and only if the regions are wisely chosen [10]. Different choices of region graphs indeed give different RBA which GBP algorithms offer various trade-offs between the complexity and the accuracy. Besides, regions chosen regardless of the factor graph topology surely lead to very poor results. How to optimally choose regions for a GBP algorithm, though, remains an open research problem.

In this work, we focus on Low-Density Parity-Check (LDPC) codes [11], [12]. These codes are capacity-achieving codes and usually decoded by BP. When approaching high signal-to-noise ratios (SNRs), LDPC codes though reveal an error-floor [13] due to special error events, the trapping sets, that are not corrected by BP. Studies were conducted to lower floors. The most of them were dedicated to codes design [14], [15] but a very few was oriented toward decoding strategies [16], [17]. Here, we propose a novel decoding strategy based on the GBP. No dedicated region-graph has been proposed in the literature for LDPC codes. This paper contributes to the choice of regions by highlighting constructions that are likely to give good results for LDPC codes. We propose to select regions from trapping sets, each one being equivalent to a local loopfree region-graphs on which GBP optimally performs.

In addition, GBP is known to be unstable [18], that degrades its performance. We then append to this paper a study on the damping of update equations of GBP messages to stabilize them. We give a profile of the damping evolution along the SNR to obtain better convergence and accuracy of GBP.

The outline for the paper is as follows. In section II, we review the factor graphs, the rules of the RBA and the basics to construct a region-graph. In section III, we introduce our method to associate a well-suited region-graph to trapping sets of LDPC codes. We detail in section IV how we damp GBP to stabilize its update equations. We end by section V where we present numerical results on the Binary Symmetric Channel (BSC) and the Additive White Gaussian Channel (AWGNC).

II. Preliminaries

A. Factor graphs

Consider a set of N binary variables $\mathbf{X} = \{X_1, \ldots, X_N\}$, and a set of M non-negative functions $\mathbf{F} = \{f_1, \ldots, f_M\}$. A function f_a softly constrains a few variables $\mathbf{X}_a \subseteq \mathbf{X}$, such that the prior probability function is:

$$P(\mathbf{X} = \mathbf{x}) \triangleq p(\mathbf{x}) = \frac{1}{Z} \prod_{a=1}^{M} f_a(\mathbf{x}_a) \tag{1}$$

where $Z = \sum_{\mathbf{x}} \prod_{a=1}^{M} f_a(\mathbf{x}_a)$ is a normalization constant. A factor graph is a bipartite graph that represents the factorization (1) with a variable node for each variable and a function node for each function. We draw an edge $e_{ia} \in E$ between

nodes X_i and f_a if and only if X_i is an argument of f_a. The factor graph is then denoted by $\mathcal{G} = (\mathbf{X} \cup \mathbf{F}, E)$.

A Tanner graph of a binary LDPC code $\mathcal{C}$ is a factor graph where functions are only valued in $\{0, 1\}$. They represent the parity-check equations of $\mathcal{C}$, which function nodes are called check nodes. A word $\mathbf{x}$ is called a codeword when all parity-check equations are satisfied, *i.e.* $p(\mathbf{x}) > 0$.

B. Inference

Consider N independent observations $y_1, \ldots, y_N$. The Bayesian rule provides the joint probability function:

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{y}|\mathbf{x})p(\mathbf{x}) = \frac{1}{Z} \prod_{i=1}^{N} p_i(y_i|x_i) \prod_{a=1}^{M} f_a(\mathbf{x}_a) \quad (2)$$

The inference problem consists in finding the state $\hat{\mathbf{x}}$ that maximizes $p(\mathbf{x}, \mathbf{y})$. Computing (2) for all 2^N states is an intractable job as N often reach very large values, *e.g.* $N = 64800$ in DVB-S2 standard [19]. We need approximation methods to approach the solution, as the region-based approximation that we describe in the next parts.

C. Region-based approximation

Region-Based Approximation (RBA) introduced in [8] approaches (2) with a factorized distribution. The factors are marginal functions $\{b_r(\mathbf{x}_r)\}_{r,\mathbf{x}_r}$, that are called beliefs, over subgraphs of $\mathcal{G}$, called regions. We denote any region by $r = (\{\mathbf{X}_r \subseteq \mathbf{X}\} \cup \{\mathbf{F}_r \subseteq \mathbf{F}\}, \{E_r \subseteq E\})$. A set of regions $\mathcal{R}$ is valid for a given factor graph $\mathcal{G}$ if and only if:

- (C1) any region that contains f_a also contains $\mathbf{X}_a$,
- (C2) any node in $\mathcal{G}$ is contained in one region, at least.

$\mathcal{R}$ then approximates (2) by:

$$b(\mathbf{x}, \mathbf{y}) = \prod_{r \in \mathcal{R}} b_r^{c_r}(\mathbf{x}_r, \mathbf{y}_r) \quad (3)$$

Any region function $b_r(\mathbf{x}_r)$ is weighted by a counting number c_r to comply with the Bayesian rule. Consider an example with function nodes f_a, f_b, f_c and four variable nodes such that $\mathbf{X}_a = \{X_1, X_2\}, \mathbf{X}_b = \{X_2, X_3\}, \mathbf{X}_c = \{X_2, X_4\}$. In the factorizations ($\mathbf{y}$ is implicit):

$$b_\alpha(x_1, x_2, x_3, x_4) = b_a(x_1, x_2) b_b(x_2, x_3) b_c(x_2, x_4) \quad (4)$$

$$b_\beta(x_1, x_2, x_3, x_4) = \frac{b_{ab}(x_1, x_2, x_3) b_c(x_2, x_4)}{b_2(x_2)} \quad (5)$$

b_α is incorrect because X_2 is counted twice too many on the right side. On the contrary, b_β is valid as X_2 is well-balanced between the numerator and the denominator. A factorization is then valid if and only if:

$$\forall X_i \in \mathbf{X}, \sum_{\mathbf{X}_r \ni X_i} c_r = 1, \quad \forall f_a \in \mathbf{F}, \sum_{\mathbf{F}_r \ni f_a} c_r = 1 \quad (6)$$

An RBA is graphically represented by a region-graph. The nodes of the region-graph are the regions of a valid set $\mathcal{R}$. The edges are given by the relationships between the regions as we explain here after. Consider two regions s, r of $\mathcal{R}$. We define a restrictive inclusion law $s \prec r$ if and only if $s \subset r$, *i.e.* $\mathbf{X}_s \subseteq \mathbf{X}_r, \mathbf{F}_s \subseteq \mathbf{F}_r, E_s \subseteq E_r$, and no region t could be found in $\mathcal{R}$ s.t. $s \subset t \subset r$. Regions are then linked as:

- $s \prec r \equiv s$ belongs to $\mathcal{E}_r$ the set of children of r, r belongs to $\mathcal{P}_s$ the set of parents of s,
- $s \subset r \equiv s$ belongs to $\mathcal{D}_r$ the set of descendants of r, r belongs to $\mathcal{A}_s$ the set of ancestors of s,
- $r \cup \mathcal{D}_r$ is the family of r denoted by $\mathcal{F}_r$.

Two regions r, s, with $s \in \mathcal{E}_r$, draw a directed edge from the node r to the node s in the region-graph.

For a given factor graph, we may find numerous valid sets of regions that build numerous region-graphs. All region-graphs offer distinct approximations (3) of various accuracies. In the next parts, we then focus on the choice of $\mathcal{R}$.

D. Region-graph construction

The first step of RBA is to cluster the factor graph: 1) we determine loopy subgraphs in $\mathcal{G}$ that we want to neutralize, 2) we associate to each of these subgraphs a region, called a cluster, in a set $\mathcal{R}$. If $\mathcal{R}$ is valid according to (C1) and (C2), it defines the first and highest generation of the region-graph. However, $\mathcal{R}$ is incomplete because it does not define a valid factorization yet (6). We complete $\mathcal{R}$ building other generations of regions by this algorithm:

3) we search for the largest children that intersect regions of the upper generation,
4) we associate to each of these children a region in the current generation,
5) each children is then connected by directed edges to its parents of the upper generation.

We repeat steps 3)-4)-5) while last built regions intersect. We exhibit in Fig.1 a factor graph and a region-graph in Fig.2.

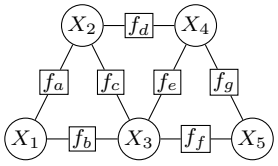


Figure 1. An example of a factor graph

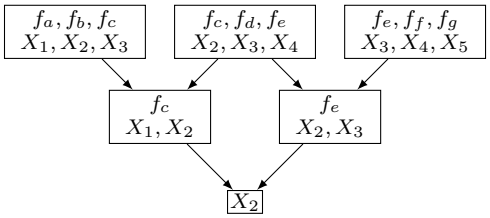


Figure 2. A region graph for the factor graph in Fig.1. The highest regions are the clusters.

E. Generalized Belief Propagation

Now the region-graph is built, we compute the marginal distributions $\{b_r(\mathbf{x}_r)\}_{r,\mathbf{x}_r}$. This job is done with Generalized Belief Propagation (GBP). GBP is an iterative algorithm

that passes messages between regions of the region-graph. According to [8], [9], for two regions r, s s.t. $s \in \mathcal{E}_r$, the message from r to s, at any iteration k, is :

$$m_{rs}^{(k)}(\mathbf{x}_s, \mathbf{y}_s) = \frac{\sum_{\mathbf{x}_r \cup \mathbf{x}_s} \beta_r(\mathbf{x}_r, \mathbf{y}_r) \prod_{\substack{u \in \mathcal{R} \setminus \mathcal{F}_r \\ v \in \mathcal{F}_r \setminus \mathcal{F}_s}} m_{uv}^{(k-1)}(\mathbf{x}_v, \mathbf{y}_v)}{\beta_s(\mathbf{x}_s, \mathbf{y}_s) \prod_{\substack{u \in \mathcal{D}_r \setminus \mathcal{F}_s \\ v \in \mathcal{D}_s}} m_{uv}^{(k)}(\mathbf{x}_v, \mathbf{y}_v)} \quad (7)$$

where:

$$\beta_r(\mathbf{x}_r, \mathbf{y}_r) = \prod_{X_i \in \mathbf{X}_r} p_i(y_i|x_i) \prod_{f_a \in \mathbf{F}_r} f_a(\mathbf{x}_a) \quad (8)$$

Once all messages have been computed, we obtain the belief of the region r:

$$b_r^{(k)}(\mathbf{x}_r, \mathbf{y}_r) = \beta_r(\mathbf{x}_r, \mathbf{y}_r) \prod_{\substack{v \in \mathcal{F}_r \\ u \in \mathcal{P}_v \setminus \mathcal{F}_r}} m_{uv}^{(k)}(\mathbf{x}_v, \mathbf{y}_v) \quad (9)$$

In papers [8] and [10], authors perform RBA for factor graphs that represent spin glasses: degree of f_a's is two (pairwise interactions). Results are encouraging but hardly applicable to LDPC codes which check nodes degree is larger than two. Authors in [20] were the first to experiment GBP on LDPC codes, they do not consider their topologies but the dependence on partial-response channels. In [18] is taken into account the codes structure. Promising results encourage us to go deeper in this direction. We describe in the next section our specific RBA for LDPC codes.

III. Tanner graph clustering

In this section we apply RBA to decode LDPC codes. We stress that RBA is aimed at neutralizing specific patterns in factor graphs. We then focus on particular BP failures that we want to treat: the trapping sets. After that, we describe the method we use to absorb and neutralize them.

A. Trapping sets

A trapping set TS(a, b) is a structure of a variable nodes which induced subgraph has b odd-degree check nodes [21], see in Fig.3 a TS$(5, 3)$.

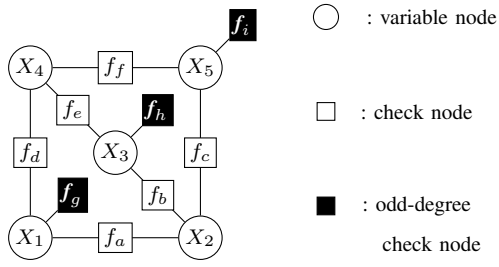


Figure 3. Tanner graph of a TS$(5, 3)$

In our work, we consider the Tanner code [22], an LDPC code of length $N = 155$, which variable nodes degree is $d_v = 3$ and check nodes degree is $d_c = 5$. This code exactly contains 155 TS$(5, 3)$ and any check node belongs to five of them. The good point of this code is that trapping sets make BP results significantly poor to easily observe improvements.

Trapping sets of any size should be taken as relevant bases to run GBP on LDPC codes. Each trapping set of arbitrary size (a, b) needs a deep study to be neutralized [23]. Here we focus on TS$(5, 3)$, our goal being to show that RBA is able to decrease their influence on the decoding process. As numerous region-graphs might be created from the Tanner code, the challenge is then to present a region-graph which GBP is more efficient than BP.

B. Novel clustering

The Tanner graph is a particular region-graph made with:

- the first generation of M clusters that contain each one a distinct check node $f_a \in \mathbf{F}$ and its neighborhood $\mathbf{X}_a$,
- the second and last generation of N regions each one having one distinct variable node $X_i \in \mathbf{X}$.

In [10], good results show that appending upper clusters to a region-graph may improve the performance of RBA. We apply this method by inserting upper clusters in the Tanner graph. The basic idea that consists in making a cluster from a whole trapping set is unpractical. In fact, the computation complexity of a GBP message grows with the size of the transmitter region, see (7). In the Tanner code, the size of a current cluster is d_c whereas an upper cluster made of a TS$(5, 3)$ would contain $6(d_c - 2) + 3(d_c - 1) + 5 = 35$ variable nodes, that is practically unreasonable. Instead, we split a TS$(5, 3)$ into several small clusters. This may improve GBP performance if it is in accordance with our rule:

Local loopfree construction: *The region-graph $\mathcal{R}_{TS}$ of a trapping set must be loopfree to have an optimal GBP on $\mathcal{R}_{TS}$.*

We justify this construction by stressing that trapping sets are very harmful for BP when the SNR is high. In this context, the subgraphs induced by trapping sets rarely all contain harmful errors in the same channel noise realization, very few of them are simultaneously concerned with BP failures. As a consequence, RBA has not to take into account the whole Tanner graph topology but only local structures. Each one describes an error event that might be eventually corrected by GBP. We represent in Fig.4 how we split a TS$(5, 3)$ to obtain a loopfree region-graph shown in Fig.5.

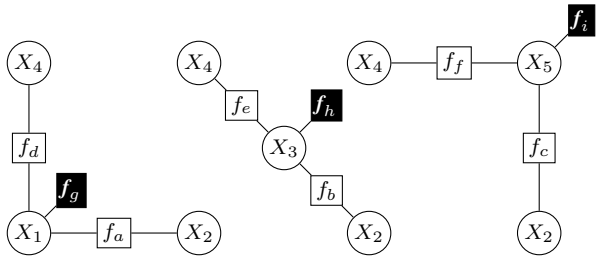


Figure 4. Split of a TS$(5, 3)$

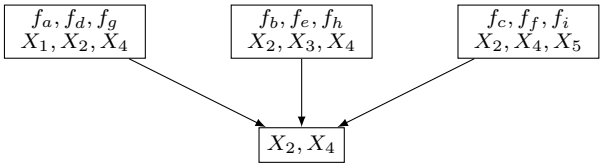


Figure 5. Region-graph resulting from the split of a TS$(5, 3)$

Now we have a well-suited region-graph to deal with TS$(5, 3)$, we are about to run GBP on the Tanner code. However this decoder is not stable at this time. In the next section, we propose a solution to stabilize and improve it in terms of computation time and BER performance.

IV. Damped GBP

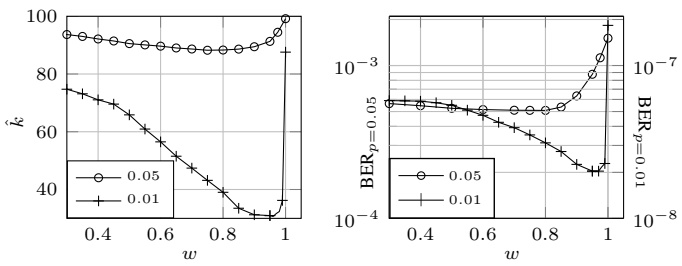
In [8] is noticed that GBP hardly converges. This may be due to the computation of a message (7) that involves at once numerous other messages from two iterations, contrary to BP that is "more local". Then, GBP cannot converge as easily as BP even for harmless channel realizations. Authors in [8] partly solve this instability by uniformly blending, at any iteration k, any message $m_{rs}^{(k-1)}$ from the former iteration with the update (7), denoted by F_{rs}. However, nothing ensures that this mixture is optimal. In [9], authors propose another mixture modeled by a decreasing damping factor $\{w_k\}_k$:

$$\forall k \in \{1, \dots, K\}, \quad m_{rs}^{(k)} = w_k F_{rs} + (1 - w_k) m_{rs}^{(k-1)} \quad (10)$$

with $w_K = 0$, K being the last iteration given by the experimenter. This process stabilizes GBP by forcing its convergence, but it does not take into account the performance in terms of the BER. In [18] is given a first study on this problem, considering different evolutions of the damping factor. Here, we show new results considering a constant damping factor $w \in [0; 1]$. For any couple $r, s \in \mathcal{E}_r$:

$$\forall k \in \{1, \dots, K\}, \quad m_{rs}^{(k)} = w F_{rs} + (1 - w) m_{rs}^{(k-1)}. \quad (11)$$

We do not have any mathematical method to get the factor w^* that stabilizes GBP and offers the best BER. We obtain w^* through a brute force algorithm: we make vary w and we compute, over numerous channel realizations, the average BER and $\hat{k}$, the average number of iterations needed either to converge or to reach a codeword. We perform this for moderate and low noise powers on the BSC, see Fig.6.



(a) Convergence (b) Performance

Figure 6. Damped GBP on the Tanner code with BSC, $p \in \{0.01, 0.05\}$

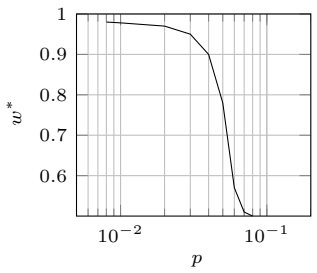


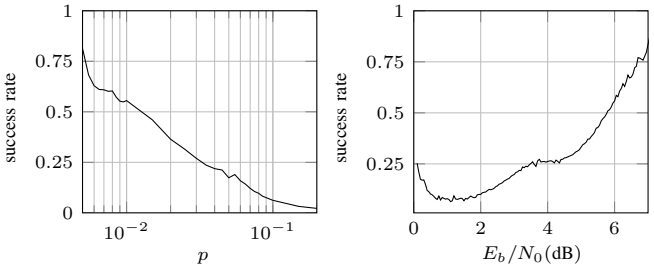
Figure 7. Optimal damping factor for GBP on the Tanner code with a BSC

On one hand, a no damped GBP, *i.e.* $w = 1.0$, is clearly outperformed by any damped GBP. On the other hand, F_{rs} must not be completely dominated by the memory, as low w values increase BER and $\hat{k}$. Besides, we see that the uniform mixture ($w = 0.5$) is not optimal, especially when channel noise is low. As an example, at $p = 0.01$, $\mathrm{BER}(0.95) = 0.2 \times \mathrm{BER}(0.5)$ and $\hat{k}(0.95) = 0.42 \times \hat{k}(0.5)$. We finally stress that the best strategy is to softly increase w as the noise gets weaker. We show this strategy in Fig.7 where we see the average value w^* against the crossover probability p.

From now on, we are able to run GBP expecting relevant performance in terms of $\hat{k}$ and BER. In the next section we present numerical results of our work.

V. Results

Here, we present experimental results of GBP run on the Tanner code. GBP may well perform as a decoder only for BP failures, in particular when related to TS$(5, 3)$. GBP is then unnecessary for error events correctly decoded by BP. Thus, we build a Hybrid Decoder (HD) that runs GBP only if BP fails, *i.e.* neither it converges nor it reaches a codeword. We consider the BSC and the AWGNC. HD and BP are run for $K = 100$ iterations. GBP is damped with a various weight w according to Fig.7 on BSC and a similar profile on AWGNC. All next quantities are averaged over 10^{12} channel realizations.



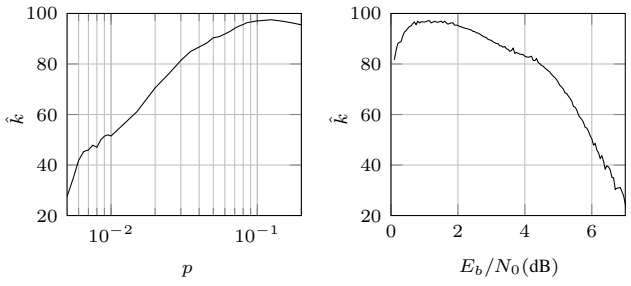
(a) BSC (b) AWGNC

Figure 8. GBP success rate on the Tanner code

We evaluate the success rate of GBP defined as the ratio between the number of GBP decodings that succeed when BP fails, and the number of BP failures. As shown in Fig.8, GBP is gradually more successful as the SNR is increased. We then confirm that TS$(5, 3)$ wield much influence for weak noise and that splitting TS$(5, 3)$ entails a decline in this influence.

We exhibit in Fig.9 the computation time modeled by the average value of $\hat{k}$ against the SNR. When error events are highly weighted for large noise powers, GBP is not fast as the

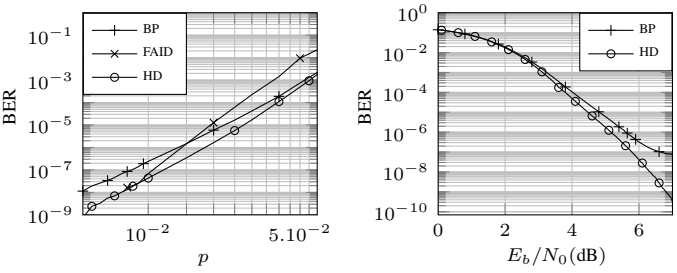
region-graph cannot deal well with such situations. In contrast, when error events are low weighted, we see a clear decreasing in $\hat{k}$. GBP then performs locally, treating well each TS$(5,3)$ in a loopfree region-graph.



(a) BSC (b) AWGNC

Figure 9. Number of iterations of BP and GBP on Tanner code

This observation is confirmed in Fig.10 where we represent the average BER of HD and BP as functions of the noise power. The gap between performance of HD and BP widens as the SNR is increased. In other words, trapping sets TS$(5,3)$ wield less influence on HD. The error-floor of HD is then lower than that of BP. We can observe this phenomenon on the BSC for $p \leq 3.10^{-2}$ and on the AWGNC for $E_b/N_0 \geq 6.0$dB.



(a) BSC (b) AWGNC

Figure 10. Hybrid decoder BER on Tanner code

In addition, we append to Fig.10(a) the average BER obtained from Finite Alphabet Iterative Decoders (FAID) [17]. We observe that HD reaches similar performance as FAID. The diversity of RBA remains in its capacity to consider any trapping set of various size. We hope then that this diversity will be a major asset to offer an even more competitive GBP.

VI. Conclusion

In this paper we have dealt with the choice of regions in RBA to run an efficient GBP for LDPC codes. We presented a new approach where regions are made with split trapping sets and added bottom-up to the Tanner graph. GBP then becomes locally loopfree on trapping sets.

We also elevate the GBP instability by a new damping scheme, improving the computation time and the BER. The experimental results have shown that RBA can be applicable to LDPC codes, which GBP outperforms BP in the error-floor.

We stressed that RBA is dedicated to deal with specific BP failures, *i.e.* other trapping sets of different sizes need other region-graphs to be neutralized. Designing other RBAs for each trapping set would offer even better improvements.

References

[1] F.R. Kschischang, B. J. Frey and H.A. Loeliger, "Factor Graphs and the Sum-Product Algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, 2001.

[2] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[3] K.P. Murphy, Y. Weiss and M.I. Jordan, "Loopy belief propagation for approximate inference: an empirical study," in *Proc. Uncertainty in Artificial Intelligence*, 1999, pp. 467–475.

[4] T. Heskes, "Convexity arguments for efficient minimization of the Bethe and Kikuchi free energies," *Journal of Artificial Intelligence Research*, vol. 26, no. 1, pp. 153–190, 2006.

[5] A.L. Yuille and A. Rangarajan, "The Concave-Convex Procedure (CCCP)," *Neural Computation*, vol. 15, no. 4, pp. 915–936, 2003.

[6] H. Wymeersch, F. Penna and V. Savic, "Uniformly Reweighted Belief Propagation: A Factor Graph Approach," in *Proc. IEEE International Symposium on Information Theory*, 2011, pp. 2000–2004.

[7] T.G. Roosta, M.J. Wainwright and S.S. Sastry, "Convergence Analysis of Reweighted Sum-Product Algorithms," *IEEE Transactions on Signal Processing*, vol. 56, no. 9, pp. 4293–4305, 2008.

[8] J.S. Yedidia, W.T. Freeman and Y. Weiss, "Constructing free energy approximations and Generalized Belief Propagation algorithms," *IEEE Transactions on Information Theory*, vol. 51, no. 7, pp. 2282–2312, 2005.

[9] P. Pakzad and V. Anantharam, "Estimation and marginalization using Kikuchi approximation methods," *Neural Computation*, vol. 17, no. 8, pp. 1836–1873, 2005.

[10] M. Welling, "On the Choice of Regions for Generalized Belief Propagation," in *Proc. Uncertainty in Artificial Intelligence*, 2004, pp. 585–592.

[11] R.G. Gallager, "Low-Density Parity-Check Codes," Ph.D. dissertation, MIT, 1963.

[12] D. J. C. MacKay, "Good Error-Correcting Codes Based on Very Sparse Matrices," *IEEE Transactions on Information Theory*, vol. 45, no. 2, pp. 399–431, 1995.

[13] T.J. Richardson, "Error floors of LDPC codes," in *Proc. 41st Annual Allerton Conference on Communication, Control and Computing*, 2003, pp. 1426–1435.

[14] D.V. Nguyen, S.H. Chilappagari, M.W. Marcellin and B. Vasic, "On the Construction of Structured LDPC Codes Free of Small Trapping Sets," *IEEE Transactions on Information Theory*, vol. 58, no. 4, pp. 2280–2302, 2012.

[15] J. Xu, L. Chen, I. Djurdjevic, S. Lin and K.Abdel-Ghaffar, "Construction of regular and irregular LDPC codes: geometry decomposition and masking," *IEEE Transactions on Information Theory*, vol. 53, no. 1, pp. 121–134, 2007.

[16] Y. Han and W. E. Ryan, "Low-Floor Decoders for LDPC Codes," *IEEE Transactions on Communications*, vol. 57, no. 6, pp. 1663–1673, 2009.

[17] L. Danjean, D. Declercq, S. K. Planjery, B. Vasic, "On the selection of finite alphabet iterative decoders for LDPC codes on the BSC," in *Proc. IEEE Information Theory Workshop*, 2011, pp. 345–349.

[18] J.-C. Sibel, S. Reynal and D. Declercq, "A novel region graph construction based on trapping sets for the Generalized Belief Propagation," in *Proc. IEEE Conference on Communication Systems*, 2012, pp. 305–309.

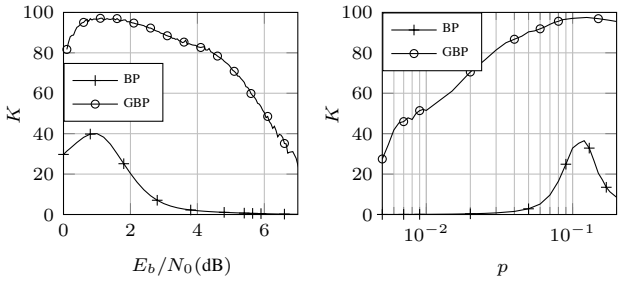
[19] A. Morello and V. Mignone, "DVB-S2: The Second Generation Standard for Satellite Broad-band Services," *Proc. IEEE*, vol. 94, pp. 210–227, 2006.

[20] P. Pakzad and V. Anantharam, "Kikuchi Approximation Method for Joint Decoding of LDPC Codes and Partial-Response Channels," *IEEE Transactions on Communications*, vol. 54, no. 7, pp. 1149–1153, 2006.

[21] S. Sankaranarayanan, S. K. Chilappagari, R. Radhakrishnan and B. Vasic, "Failures of the Gallager B Decoder: Analysis and Applications," in *Proc. Information Theory and Applications Workshop UCSD*, 2006.

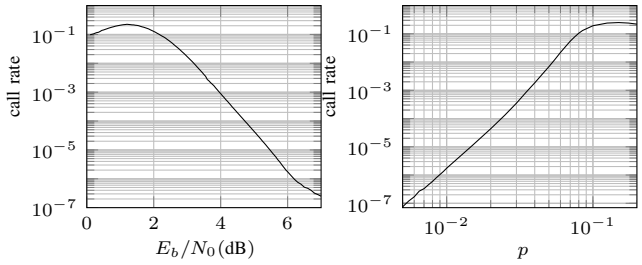
[22] R.M. Tanner, R. Michael, D. Sridhara and T. Fuja, "A Class of Group-Structured LDPC Codes," 2001.

[23] B. Vasic, S.K. Chilappagari, D.V. Nguyen and S.K. Planjery, "Trapping set ontology," in *Proc. Allerton Conference on Communication, Control and Computing*, 2009, pp. 1–7.



(a) AWGN (b) bsc

Figure 11. Number of iterations of BP and GBP on Tanner code. For GBP, we only consider the simulations for which BP fails, explaining why K_{GBP} is greater than K_{BP}



(a) AWGN (b) bsc

Figure 12. Call rate of GBP: number of calls to GBP (i.e. number of BP failures) divided by number of simulations

ACTIVITY

As a decoder is passing messages along the edges of a graph, trapping sets manifest by making the iterative output estimate $\hat{\mathbf{x}}^{(k)}$ oscillate. The amplitude of these oscillations somehow refers to the intensity of the trapping sets influence, or simply the trapping sets activity. To define this activity, we first define the K-variance of X_i:

$$\sigma_i^2 = \frac{1}{K}\sum_{k=1}^{K}\left(\hat{x}_i^{(k)} - \bar{x}_i\right)^2 \tag{12}$$

where $\bar{x}_i =$ is the average output value of the i-th bit. The activity of a trapping set $\mathcal{T}$ that belongs to the set of all trapping sets $(5,3)$ is then:

$$\alpha_{\mathcal{T}} = \frac{1}{N_{\mathcal{T}}}\sum_{X_i \in \mathbf{X}_{\mathcal{T}}} \sigma_i. \tag{13}$$

We finally define the α-activity as the number of trapping sets which activity $\alpha_{\mathcal{T}}$ is greater than α. Fig.13 displays the mean rate of α-active trapping sets for different values of the BSC crossover probability p. We observe the hybrid decoder decreases the oscillations of trapping sets, i.e. HD appears more stable than regular BP.

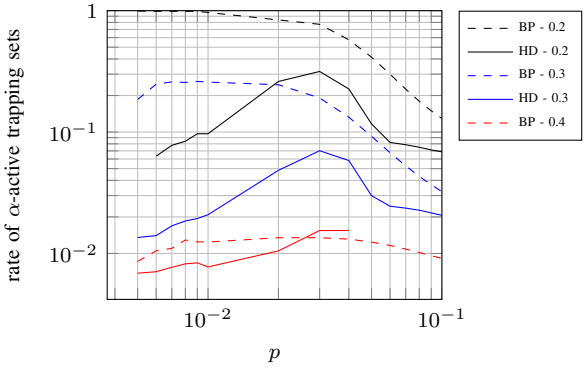


Figure 13. α-activity of trapping sets with $\alpha \in \{0.2, 0.3, 0.4\}$