Low-Complexity Approximations to Belief Propagation for LDPC Codes

Jeremy Thorpe

October 31, 2002

1 Summary

The standard belief propagation algorithm that is used to decode Low-Density-Parity-Check codes defines real-valued messages that are passed along edges in a graph. The standard way to simulate this algorithm is to use a very accruate representation of the real numbers, such as a floating-point numbers, to store the value of each message. However, for very high-speed decoders, it is clear that the high complexity associated with computing and storing such numbers is to be avoided if possible. Indeed, Gallager's Algorithm A[1] can be seen as a single bit approximation to belief-propagation algorithm. This paper explores some of the ground which lies between the two extremes. In particular, we investigate semi-analytically (via density evolution) and through simulation several rules having message size between 1 and 4 bits.

2 Background

An LDPC code is characterized by a bipartite graph $G = (C, V, E)$, where $V, |V| = r$ is a set of variable nodes $C, |C| = n$ is a set of check nodes, and E is a set of edges that have one endpoint in C and one endpoint in V. A vector $X \in F(2)^n$, indexed by the elements of V, is a codeword if and only if $\sum_{v|c} X_v = 0 \forall c \in C$. The notation $v|c$ means any $v \in V$ connected to c via an edge $e \in E$.

If we transmit a codeword X over a memoryless channel and receive Y. The belief-propagation (BP) algorithm estimates X given Y. BP is an iterative algorithm in which messages are passed in each direction on each edge of E. In particular, messages denoted m_v which are the inputs to the decoder are calculated based on the channel outputs and channel statistics. The messages $m_{c,v}^{(l)}$ are passed from v to c at (the first half of) iteration l, and similarly messages denoted $m_{c,v}^{(l)} \in \mathbb{R}$ are passed from c to v at (the second half of) iteration l.

The input messages to the BP algorithm are calculated from the channel outputs:

$$m_v = \log\left(\frac{p\left(Y_v|X_v=0\right)}{p\left(Y_v|X_{\ v}=1\right)}\right)$$

At iteration 0, each message $m_{v,c}^{(0)}$ passed along each edge (v,c) is simply equal to m_v:

$$m_{v,c}^{(0)} = m_v$$

At each iteration l, each message $m_{c,v}^{(l)}$ passed along edge (v,c) is calculated in terms of the messages $m_{v',c}^{(l)}$:

$$m_{c,v}^{(l)} = \sum_{v'|c\neq v} t\left(m_{v',c}^{(l)}\right)$$

At each iteration $l>0$, each message $m_{v,c}^{(l)}$ are calculated in terms of the messages $m_{c',v}^{(l-1)}$:

$$m_{v,c}^{(l)} = m_v + \sum_{c'|v\neq c} t\left(m_{c',v}^{(l-1)}\right)$$

After a sufficient number L of iterations, we estimate each symbol X_v as:

$$X_v = \begin{cases} 0, & m_v + \sum_{c'|v} t\left(m_{c',v}^{(L)}\right) > 0 \\ 1, & \text{otherwise} \end{cases}$$

The function t is defined as:

$$t(x) = \log\left(\frac{1-e^x}{1+e^x}\right)$$

Note that for $x<0$, $t\left(x\right)$ includes the term $\log(-1)$, which can be treated as a special symbol.

Messages $m_{v,c}^{(l)}$ are said to be in *reliability* domain, which is in a certain sense the additive domain for the variable nodes, while the messages $m_{c,v}^{(l)}$ are in the *unreliability* domain, the additive domain for check nodes.

3 Quantized Belief Propagation

We define a new algorithm which approximates BP, quantized belief propagation (QBP). QBP essentially approximates the BP algorithm as it is stated in the previous section; it operates in each of the additive domains. Whereas the BP algorithm is defined by a set of messages m, QBP will be defined in terms of messages m'. While in the standard BP algorithm defines messages $m_v \in \mathbb{R}$, $m_{v,c}^{(l)} \in \mathbb{R}$, and $m_{c,v}^{(l)} \in \mathbb{R}$, in QBP we define messages $m_v \in \mathcal{M}_{ch} = \{\pm 1, \pm 2 ... \pm \frac{q_{ch}}{2}\}$, $m_{v,c}^{\prime(l)} \in \mathcal{M}_v = \{\pm 1, \pm 2 ... \pm \frac{q_v}{2}\}$, and $m_{c,v}^{\prime(l)} \in \mathcal{M}_c = \{\pm 1, \pm 2 ... \pm \frac{q_c}{2}\}$. For simplicity, we formulate QBP for

We first give an abstract quantization rule $Q_\tau : X \mapsto \{\pm 1, \pm 2 ... \pm \frac{q}{2}\}$, where $\tau = \left\{\tau_1, ..., \tau_{\frac{q}{2}-1}\right\}$, $\tau_i \in \mathbb{R}^+$ is a set of thresholds with $\tau_i < \tau_{i+1}$.

$$Q_\tau(x) = \begin{cases} -\frac{q}{2}, & x < -\tau_{\frac{q}{2}-1} \\ -\arg\min_i\{\tau_i > -x\}, & -\tau_{\frac{q}{2}-1} \leq x < 0 \\ \arg\min_i\{\tau_i > x\}, & 0 \leq x \leq \tau_{\frac{q}{2}-1} \\ \frac{q}{2}, & \tau_{\frac{q}{2}-1} < x \end{cases}$$

The reconstruction function $\phi : \{\pm 1, \pm 2.. \pm \frac{q}{2}\} \to \mathbb{N}$ is an anti-symmetric function which is characterized by its values on $\{1, 2 ... \frac{q}{2}\}$.

$$\phi(-i) = -\phi(i)$$

The input to the QBP decoder are defined in terms of the input to the ideal BP decoder and the parameter τ_{ch}:

$$m'_v = Q_{\tau_{ch}}(m_v)$$

At iteration 0, the message passed along each edge (v, c) is equal to v's input message:

$$m'^{(0)}_{v,c} = Q_{\tau_v}(\phi_v(m_v))$$

Each message $m^{(l)}_{c,v}$ are calculated with respect to several of the messages $m^{(l)}_{v',c}$ as:

$$m'^{(l)}_{c,v} = Q_{\tau_c}\left(\sum_{v'|c\neq v} \phi_c\left(m^{(l)}_{v',c}\right)\right)$$

On all iterations $l > 0$, the messages $m^{(l)}_{v,c}$ are calculated as:

$$m'^{(l)}_{v,c} = Q_{\tau_v}\left(\phi_{ch}\left(m_v\right) + \sum_{c'|v\neq c} \phi_v\left(m^{(l-1)}_{c',v}\right)\right)$$

After a sufficient number L of iterations, we estimate the symbol X_v as:

$$X_v = \begin{cases} 0, & \phi_{ch}\left(m_v\right) + \sum_{c'|v} \phi_v\left(m^{(l-1)}_{c',v}\right) > 0 \\ 1, & \text{otherwise} \end{cases}$$

Note that if ϕ_{ch} and ϕ_v are such that the sum in the previous equation can be exactly 0, then the algorithm cannot be both symmetric. A simple way to avoid this is for $\phi_{ch}\left(i\right)$ to be odd for all i and $\phi_v\left(i\right)$ to be even for all i.

4 QBP Rules for the $(3, 6)$ Regular LDPC Ensemble

There are a number of ways to test the goodness of different QBP algorithms. A simple and direct way is to simulate the algorithm on particular codes for

a given channel. Another way is density evolution, by which we calculate the fractions of edges transmitting each message, assuming infinte block-length. This method essentially calculates the asymptotic performance of the code as the code length n approaches infinity. This has the advantage of being quite fast to compute, but the disadvantage that it may fail to predict performance when there are loops in the graph, as will be seen in seciton 5.

In this section, density evolution is used to predict the performance of QBP for the class of regular $(3, 6)$ LDPC codes. The results are characterized by a value SNR^* for which if $SNR < SNR^*$ we have bad performance, and for which if $SNR > SNR^*$ we have bit error given by E, which may or may not be equal to 0. if $E = 0$, we say the algorithm has no error floor, otherwise it has an error floor. It is not necessarily true that algorithms with higher values of q_{ch}, q_v, and q_c have no error floor if the same holds for an algorithm with lower values. In particular, the simplest algorithm, with $q_{ch} = q_v = q_c = 2$ has no error floor, while more complex algorithms do have error floors.

The following table summarizes the best known QBP rules for a range of interesting values of q_{ch}, q_v, and q_c.

Name	I[1]	II	III	IV	V[2]
q_{ch}	2	4	4	8	8
q_v	2	4	4	4	8
q_c	2	4	4	4	8
τ_{ch}[3]	∞	1.76	1.4	1.4	1.4
ϕ_{ch}	(1)	(1, 3)	(3, 15)	(1, 3, 5, 7)	(3, 9, 15, 21)
τ_v	()	(2)	(7)	(2)	(6, 12, 18)
ϕ_v	(1)	(1, 3)	(3, 11)	(1, 3)	(2, 6, 8, 12)
τ_c	()	(5)	(5)	(5)	(7, 11, 24)
ϕ_c	(1)	(1, 2)	(1, 2)	(1, 2)	(1, 2, 6, 21)
SNR^*	4.896	2.248	1.955	1.689	1.443
E	0	0	$3 \cdot 10^{-3}$	$7 \cdot 10^{-3}$	$5 \cdot 10^{-4}$

Name	VI	VII	Ideal
q_{ch}	8	16	∞
q_v	8	16	∞
q_c	8	16	∞
τ_{ch}	1.1	.66	ϵ
ϕ_{ch}	(3, 9, 15, 21)	(15, 45, 75, 105, 135, 163, 193, 245)	$\mathbb{R}$
τ_v	(6, 12, 18)	(30, 60, 90, 120, 150, 180, 210)	$\mathbb{R}$
ϕ_v	(2, 6, 12, 20)	(9, 23, 37, 53, 71, 97, 125, 167)	$\mathbb{R}$
τ_c	(5, 9, 26)	(8, 15, 28, 45, 135, 163, 193, 245)	$\mathbb{R}$
ϕ_c	(1, 2, 6, 26)	(8, 15, 28, 45, 135, 163, 193, 245)	$\mathbb{R}$
SNR^*	1.409	1.199	1.09
E	0	$3 \cdot 10^{-6}$	0

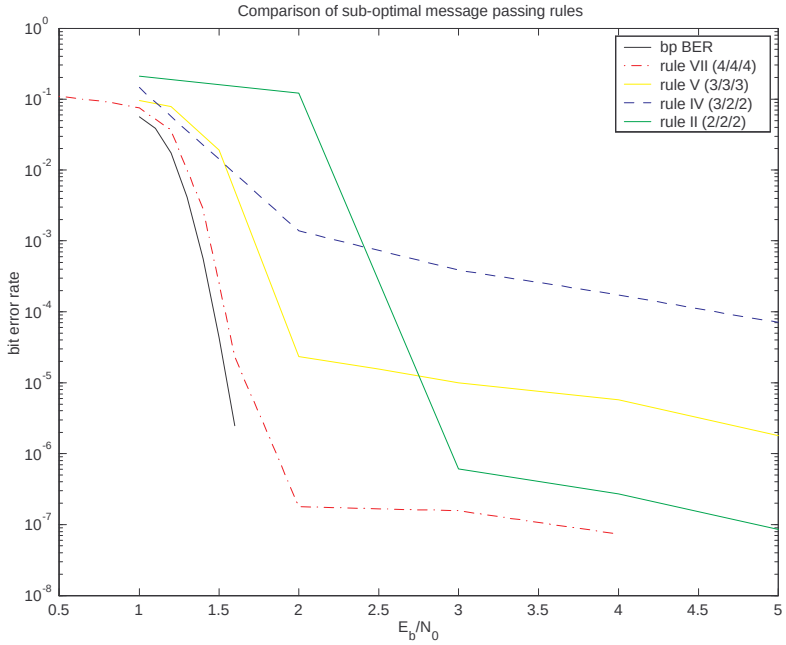
Note that rule V Corresponds precisely to the rule suggested by Richardson[2],

[1] Equivalent to Gallager's algorithm A and B for regular (3,6) LDPC codes.
[2] Equivalent to Richardson and Urbanke's rule.
[3] $\tau_{ch} = \{x, 2x, ..., (q_{ch} - 1)x\}$, x in the table.

as do the threshold and error-floor predictions.

5 Simulation Results

The following two figure shows the performance of several of the QBP algorithms applied to a regular $(3, 6)$ LDPC code.



In the waterfall region, the performance of each rule is predicted quite well from density evolution. In addition, in each instance where an error floor appears in density evolution, it appears in the simulation. However, there is in fact an error floor on rule II which is not predicted from density evolution. Preliminary analysis strongly suggests that this error floor is in fact due to loops in the graph, as opposed to tree-like configurations as in the other error floors.

In general, the error floors inherent in all of the above rules can be understood to be caused by the inability of the strongest internal messages to completely overcome the strongest messages from the channel. This suggests increasing the reliability values and thresholds τ_v and ϕ_v on the internal messages, effectively trading resolution at low reliability levels for range of expression, which would likely decrease the error floor at the expense of a threshold further from capacity. Other methods have also been suggested[2] to mitigate or eliminate error floors, such as artificially lowering the channel messages at a sufficiently late iteration. More work is needed to explore this tradeoff (and should be completed by the time of this presentation)

6 Acknowledgements

The author would like to thank Kenneth Andrews and Gill Chinn for contributing time and effort to the simulation of several of the decoding rules.

References

[1] R. G. Gallager. Low Density Parity Check Codes. Number 21 in Research monograph series. MIT Press, Cambridge, Mass., 1963

[2] T. Richardson, R. L. Urbanke. The Capacity of Low-Density Parity-Check Codes Under Message-Passing Decoding. IEEE Trans. Inform. Theory, vol. 47, Feb. 2001.