Quantized Iterative Message Passing Decoders with Low Error Floor for LDPC Codes

Xiaojie Zhang, *Member, IEEE*, and Paul H. Siegel, *Fellow, IEEE*

Abstract—The error floor phenomenon observed with LDPC codes and their graph-based, iterative, message-passing (MP) decoders is commonly attributed to the existence of error-prone substructures – variously referred to as near-codewords, trapping sets, absorbing sets, or pseudocodewords – in a Tanner graph representation of the code. Many approaches have been proposed to lower the error floor by designing new LDPC codes with fewer such substructures or by modifying the decoding algorithm. Using a theoretical analysis of iterative MP decoding in an idealized trapping set scenario, we show that a contributor to the error floors observed in the literature may be the imprecise implementation of decoding algorithms and, in particular, the message quantization rules used. We then propose a new quantization method – $(q+1)$-bit quasi-uniform quantization – that efficiently increases the dynamic range of messages, thereby overcoming a limitation of conventional quantization schemes. Finally, we use the quasi-uniform quantizer to decode several LDPC codes that suffer from high error floors with traditional fixed-point decoder implementations. The performance simulation results provide evidence that the proposed quantization scheme can, for a wide variety of codes, significantly lower error floors with minimal increase in decoder complexity.

Index Terms—Low-density parity-check (LDPC) codes, iterative message-passing decoding, sum-product algorithm, message quantization, error floors, trapping sets.

I. INTRODUCTION

THE outstanding performance of low-density parity-check (LDPC) codes and iterative, message-passing (MP) decoding algorithms [1], [2] has attracted considerable attention over the past decade and these techniques are being deployed in a growing number of practical applications. At high signal-to-noise ratio (SNR), however, LDPC codes and MP decoders may be subject to the error floor phenomenon, which manifests itself as an abrupt change in the slope of the error-rate curve. Since many important applications, such as data storage and high-speed digital communication, often require extremely low error rates, the study of error floors in LDPC codes remains of considerable practical, as well as theoretical, interest.

The error floor phenomenon is commonly attributed to the existence of certain error-prone substructures (EPSs) in a Tanner graph representation of the code. In the binary erasure channel (BEC), it has been shown that substructures known as *stopping sets* determine the error-rate performance and the observed error floor [3]. However, for general memoryless binary-input output-symmetric (MBIOS) channels such as the binary symmetric channel (BSC) and the additive white Gaussian noise channel (AWGNC), the EPSs that dominate the error floor performance have not yet been fully characterized, although some classes of EPSs have been identified and studied, such as *near-codewords* [4], *trapping sets* [5], *absorbing sets* [6], and *pseudocodewords* [7].

One common way to improve the error floor performance of LDPC codes has been to redesign the codes to have Tanner graphs with large girth and without problematic EPSs which usually consist of small number of variable nodes [8]–[10]. However, for LDPC codes that have been standardized, approaches are needed that do not modify the codes. In the literature, many modifications to the iterative MP decoding algorithms have been proposed in order to improve high SNR performance, such as averaged decoders [11], reordered decoders [12], [13], and decoders with post processing [14]–[18]. In [11], the authors noticed that the emergence of errors in EPSs is heuristically related to a sudden magnitude change in the values of certain variable nodes (VNs). Hence, it was proposed to average the messages in a belief-propagation (BP) decoder over several iterations to avoid such sudden changes and therefore slow down the convergence rate for variable nodes in a trapping set and decrease the frequency of trapping set errors. Another heuristic approach is to process messages based on the order of node reliabilities computed at each iteration [12], and it was suggested that the scheduled decoders are able to resolve some standard trapping set errors [13]. Although these general approaches are capable of improving the average error rate performance to some extent, the resulting decoders still fail on small EPSs and their effect on the error floor is not significant.

To further improve the error floor behavior, decoders that make use of the prior knowledge of some small size EPSs have been designed to reduce the decoding failures due to such EPSs. In [14] and [15], the authors proposed a post-processing decoder that matches the configuration of unsatisfied check nodes (CNs) to trapping sets in a precomputed list after conventional MP decoding has failed. The size and completeness

X. Zhang was with the Department of Electrical and Computer Engineering and the Center for Magnetic Recording Research, University of California, San Diego, La Jolla, CA 92093. He is now with Samsung Research America – Dallas, Richardson, TX 75082 (e-mail: snuzxj@gmail.com).

P. H. Siegel is with the Department of Electrical and Computer Engineering and the Center for Magnetic Recording Research, University of California, San Diego, La Jolla, CA 92093 (e-mail: psiegel@ucsd.edu).

Digital Object Identifier 10.1109/TCOMM.2013.112313.120917

of the trapping set list directly affect the performance gain of such decoders, but to obtain a complete list of small trapping sets of a given LDPC code is generally quite computationally complex. A symbol-selecting post-processing technique was also developed in [16]. It saturates the channel messages on a set of selected variable nodes at each stage after the conventional MP algorithms fails. In [17], Han and Ryan proposed a bi-mode erasure decoder that combines several problematic check nodes into a generalized constraint processor, to which a corresponding maximum a posteriori (MAP) algorithm, such as the BCJR algorithm, is then applied. Another post-processing approach that utilizes the graph-theoretic structure of absorbing sets, proposed in [18], adjusts the appropriate messages in the iterative MP decoding once the decoder enters and remains in the absorbing set of interest.

All the above approaches either change the message update rules of MP decoders or require extra processing steps after conventional MP decoding fails, both of which increase the decoding complexity relative to the original iterative MP algorithms. Moreover, the post-processing approaches that require prior knowledge of the set of EPSs causing the error floor are only effective when applied to LDPC codes whose EPSs have been carefully studied.

In fixed-point implementation of iterative MP decoding, efforts have also been made to improve the error-rate performance in the waterfall region and/or error-floor region by optimizing parameters of uniform quantization [19]–[22]. In [19], Zhao *et al.* studied the effect of message clipping and uniform quantization on the performance of the min-sum (MS) decoder in the waterfall region and heuristically optimized the number of quantization bits and the quantization step size for selected LDPC codes. In [20], a dual-mode adaptive uniform quantization scheme was proposed to better approximate the log-tanh function used in sum-product algorithm (SPA) decoding. Specifically, for magnitudes less than 1, all quantization bits were used to represent the fractional part; for magnitudes greater than or equal to 1, all bits were dedicated to the representation of the integer part. In [21], [22], Zhang *et al.* proposed a conceptually similar idea to increase precision in the quantization of the log-tanh function. Uniform quantization was applied to messages generated by both variable nodes and check nodes, but the quantization step sizes used in the two cases were separately optimized. We note, however, that none of these modified quantization schemes were primarily intended to significantly increase the saturation level, or range, of quantized messages, and in their reported simulation results, error floors can still be clearly observed.

It has been observed that the high error floors associated with certain EPSs of some LDPC codes are closely related to the saturation level imposed on messages passed in the SPA decoder. (See, for example, [23] and references cited therein.) In this work, we investigate the cause of error floors in binary LDPC codes from the perspective of the MP decoder implementation, with special attention to limitations that decrease the dynamic range of messages passed during decoding. We show that, under certain idealized assumptions, the EPSs which are commonly associated with high error floors of some LDPC codes will not trap iterative MP decoders and cause high error floors if message magnitudes

and the number of iterations are not limited. Based upon an analysis of the growth rate of messages outside an EPS in an idealized scenario, we propose a novel quasi-uniform quantization method that captures the essence of messages in different ranges of reliability. The proposed quantization method has an extremely large saturation level which prevents iterative MP decoders from being trapped by an EPS. This property, to the best of our knowledge, distinguishes it from other quantization techniques for iterative MP decoding that have appeared in the literature. With the new quantization method, it is possible to have a fixed point implementation of iterative MP decoders that achieves low error floors without an additional post-processing stage or a modification of either the decoding update rules or the graphical code representation upon which the iterative MP decoder operates. We present simulation results for min-sum decoding, SPA decoding, and some of their variants, that demonstrate a significant reduction in the error floors of four representative LDPC codes, with no increase in decoding complexity.

The remainder of the paper is organized as follows. Section II gives some notation and definitions used throughout the paper. In Section III, we analytically investigate the impact that message quantization can have on MP decoder performance and the error floor phenomenon. In Section IV, we propose an enhanced quantization method intended to overcome the limitations of traditional quantization rules. In Section V, we incorporate the new quantizer into SPA and min-sum decoding and, through computer simulation of several LDPC codes known for their high error floors, demonstrate the significant improvement in error-rate performance that this new quantization approach can afford. Section VI concludes the paper.

II. NOTATION AND DEFINITIONS

The study of the phenomenon of error floors began shortly after LDPC codes were rediscovered about a decade ago. It has been shown that the EPSs known as *stopping sets* cause the error floor in the BEC, and such EPSs have a clear combinatorial description. Enumeration of these structures makes it possible to accurately estimate the error floor [3]. However, for other MBIOS channels such as the BSC and the AWGNC, it is more difficult to establish the relationship between EPSs and error floors. In [4], it was first pointed out that the *near-codewords* caused error floors in simulations of Margulis and Ramanujan-Margulis LDPC codes on the AWGNC. The term *trapping set* proposed by Richardson [5] is operationally defined as a subset of VNs that is susceptible to errors under a certain iterative MP decoder over an MBIOS channel. Hence, this concept depends on both the channel and the decoding algorithm. In [6], the error floor is associated with some combinatorial substructures within the Tanner graph, named *absorbing sets*, which are defined independently of the channel. The absorbing sets correspond to a particular type of near-codewords or trapping sets that are stable under bit-flipping operations. All these EPSs have been believed to be the cause of error floors, and for some LDPC codes, techniques such as importance sampling used to estimate the error floor are based on the probability of decoding failures on such EPSs [5], [24]. In this section, we will show that under certain idealized assumptions about the

computation trees of variable nodes within a given EPS, as well as the correctness of variable node messages outside the EPS, conventional iterative decoders that accurately represent messages will eventually correct errors supported by the EPS.

To facilitate our discussion, we define a substructure called an *absolute trapping set* from a purely graph-theoretic perspective, independent of the channel and the decoder. Let $G = (V \cup C, E)$ denote the Tanner graph of a binary LDPC code with VNs $V = \{v_1, \ldots, v_n\}$, CNs $C = \{c_1, \ldots, c_m\}$, and edge set E.

Definition 1: A *stopping set* of size a is a configuration of a variable nodes such that the induced subgraph has no check nodes of degree-one. An (a, b) *trapping set* is a configuration of a variable nodes, for which the induced subgraph is connected and has b odd-degree check nodes. If the induced subgraph of an (a, b) trapping set does not contain a stopping set, it is called an *absolute trapping set*.

In the literature, all trapping sets of interest that contribute to the error floor of an LDPC code are of size smaller than the minimum stopping set size of the code, since otherwise the stopping sets would be the dominant contributor to the error floor [3]. Note that the requirement that an absolute trapping set contain no stopping set also implies that it must have at lease one degree-one check node. As we will discuss later in this section, these degree-one check nodes are essential because they are able to pass correct extrinsic messages into the trapping set. To the best of our knowledge, almost all trapping sets of interest in the literature are absolute trapping sets. For example, both of the well-known (5,3) trapping sets in the Tanner code of length 155, the notorious (12,4) trapping sets in the (2640,1320) Margulis code, and the (5,5) trapping set in some codes of variable-degree five are all absolute trapping sets. Unless otherwise indicated, all trapping sets referred to in this paper are absolute trapping sets, as well.

In analogy to the definition of *computation tree* in [25], we define a *k-iteration computation tree* as follows.

Definition 2: A *k-iteration computation tree* $T_k(v)$ for an iterative decoder in the Tanner graph G is a tree graph constructed by choosing variable node $v \in V$ as its root and then recursively adding edges and leaf nodes to the tree that participate in the iterative message-passing decoding during k iterations. To each vertex that is created in $T_k(v)$, we associate the corresponding node update function in G.

Let S be the induced subgraph of an (a, b) trapping set contained in G, with VN set $V_S \subseteq V$ and CN set $C_S \subseteq C$. Let set $C_1 \subseteq C_S$ be the set of degree-one CNs in the subgraph S, and let set $V_1 \subseteq V_S$ be the set of neighboring VNs of CNs in C_1. We refer to a message on an edge adjacent to VN v as a *correct* message if its sign reflects the correct value of v, and as an *incorrect* message, otherwise. Let $D(u)$ be the set of all descendants of the vertex u in a given computation tree.

Definition 3: Given a Tanner graph G and an induced subgraph S of a trapping set, a variable node $v \in V_1$ is said to be *k-separated* if, for at least one of its neighboring degree-one check nodes, $c \in C_1$, no variable node $v' \in V_S$ belongs to $D(c) \subset T_k(v)$. If every $v \in V_1$ is *k-separated*, the induced subgraph S is said to satisfy the *k-separation assumption*.

In Fig. 1(a), we show the graph of a $(4, 4)$ trapping set and some of its neighboring nodes. The set of VNs in the

trapping set is $V_S = \{v_1, v_2, v_3, v_4\}$, represented as solid black circles. The set of CNs in the trapping set is $C_S = \{c_i\}$, $1 \leq i \leq 8$. In this trapping set, every VN has a neighboring degree-one CN, i.e., $V_1 = V_S$, and $C_1 = \{c_1, c_2, c_3, c_4\}$. For example, the 3-iteration computation tree of VN v_1 is shown in Fig. 1(b). It can be verified from this computation tree that v_1 is 2-separated but not 3-separated, because $v_2 \in V_S$ is a descendant of c_1 in $T_3(v_1)$, but not in $T_2(v_1)$. It is worth noting that whether or not a trapping set satisfies the k-separation assumption depends on the Tanner graph outside the trapping set, not the trapping set itself.

We want to point out that the k-separation assumption is much weaker than the isolation assumption in [26]. The separation assumption here only applies to the VNs that have neighboring degree-one CNs in the induced subgraph S, and these neighboring degree-one CNs do not have any VNs from the trapping set as their descendants in the corresponding k-iteration computation tree. With the separation assumption, the descendants of $c \in C_1$ are separated from all the nodes in the trapping set, meaning that the incorrect messages passed in the trapping set do not affect the extrinsic messages sent towards c in the computation tree.

III. ERROR FLOORS OF LDPC CODES

A. Trapping Sets and Min-Sum Decoding

To get further insight into the connection between trapping sets and decoding failures of iterative MP decoders, we first consider a simple iterative MP decoder, the min-sum decoder, which can be viewed as a simple approximation of the sum-product algorithm. We now briefly recall the VN and CN update rules of min-sum decoding.

A VN v_i receives an input message L_i^{ch} from the channel, typically the log-likelihood ratio (LLR) of the corresponding channel output, defined as follows

$$L_i^{ch} = \log \left(\frac{\Pr(R_i = r_i \mid c_i = 0)}{\Pr(R_i = r_i \mid c_i = 1)} \right), \qquad (1)$$

where $c_i \in \{0, 1\}$ is the code bit and r_i is the corresponding received symbol.

Denote by $L_{i \to j}$ and $L_{j \to i}$ the messages sent from v_i to c_j and from c_j to v_i, respectively, and denote by $N(k)$ the set of neighboring nodes of VN v_k (or CN c_k). Then, the message sent from v_i to c_j in min-sum decoding is given by

$$L_{i \to j} = L_i^{ch} + \sum_{j' \in N(i) \setminus j} L_{j' \to i} , \qquad (2)$$

and the message from CN j to VN i is computed as

$$L_{j \to i} = \left[\prod_{i' \in N(j) \setminus i} \text{sign}(L_{i' \to j}) \right] \cdot \min_{i' \in N(j) \setminus i} |L_{i' \to j}|. \qquad (3)$$

In the initialization step, we set $L_{i \to j} = L_i^{ch}$. It can been seen from (2) and (3) that the min-sum decoding algorithm is insensitive to linear scaling, meaning that linearly scaling all input messages from the channel would not affect the decoding performance.

For the MS decoder, we can show that a trapping set does not cause decoding failure if its induced subgraph in the Tanner graph satisfies certain criteria.
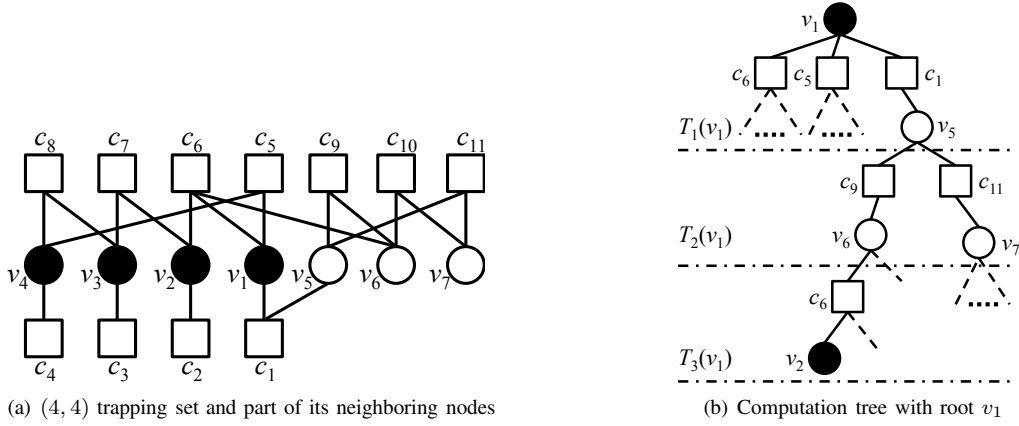
(a) $(4, 4)$ trapping set and part of its neighboring nodes

(b) Computation tree with root v_1

Fig. 1. Example of a $(4, 4)$ trapping set and its corresponding computation tree.

Theorem 1: Let G be the Tanner graph of a variable-regular LDPC code that contains a subgraph S induced by a trapping set. Assume that the channel is either a BSC or an AWGNC, and that the messages from the channel to all VNs outside S are correct. If S satisfies the k-separation assumption for sufficiently large k, then the corresponding min-sum decoder will successfully correct all erroneous VNs in S.

Proof: See Appendix A. ∎

In general, the error-rate performance of MS decoding is not as good as that of SPA decoding. However, there are several quite simple but effective ways to adjust the CN update rule of MS decoding to get comparable performance to SPA decoding. One method is *attenuated-min-sum* (AMS) decoding [27], where the magnitudes of messages are attenuated at CNs. The corresponding CN update rule of AMS is as follows

$$L_{j \to i} = \left[\prod_{i' \in N(j) \setminus i} \text{sign}(L_{i' \to j}) \right] \cdot \alpha \cdot \min_{i' \in N(j) \setminus i} |L_{i' \to j}|, \quad (4)$$

where $0 < \alpha < 1$ is the attenuation factor, which can be a fixed constant or adaptively adjusted. Another way to improve the error-rate performance of MS decoding is *offset-min-sum* (OMS) decoding, which applies an offset to reduce the magnitudes of CN output messages. The resulting CN update equation is

$$L_{j \to i} = \left[\prod_{i' \in N(j) \setminus i} \text{sign}(L_{i' \to j}) \right] \cdot \max\{ \min_{i' \in N(j) \setminus i} |L_{i' \to j}| - \beta, \ 0 \}, \quad (5)$$

where $\beta > 0$ is the offset which, like the attenuation factor, can be a fixed constant or adaptively adjusted. In some implementations, for additional simplicity, the attenuation factor or offset is set to be the same fixed constant for all CNs and all iterations [27].

Theorem 1 can be extended to both AMS and OMS decoding, where we assume that, in each iteration, all CNs use the same attenuation factor α in AMS or the same offset β in OMS.

Corollary 2: Let G be the Tanner graph of a variable-regular LDPC code that contains a subgraph S induced by a trapping set. Assume that the channel is either a BSC or

an AWGNC, and that the messages from the channel to all VNs outside S are correct. If S satisfies the k-separation assumption for sufficiently large k, then both the AMS and OMS decoder will successfully correct all erroneous VNs in S.

Proof: See Appendix B. ∎

As shown in Appendix B, the extension to AMS decoding follows easily from Theorem 1. On the other hand, the proof of the extension to the OMS decoder makes use of ideas introduced in the analysis of SPA decoding in the next subsection.

B. Trapping Sets and Sum-Product Algorithm Decoding

In this subsection, we further extend Theorem 1 to sum-product algorithm decoding. The optimality criterion in the design of the SPA decoder is symbol-wise maximum a *posteriori* probability (MAP), and it is an optimal symbol-wise decoder on Tanner graphs without cycles.

In SPA decoding, VN nodes take log-likelihood ratios of received information from the channel as initial input messages. The VN update rule is the same as that of MS decoding described in (2), which involves the summation of all incoming extrinsic messages. In the CN update rule of SPA decoding, the message sent from CN j to VN i is computed as

$$L_{j \to i} = 2\tanh^{-1} \left(\prod_{i' \in N(j) \setminus i} \tanh \frac{L_{i' \to j}}{2} \right). \quad (6)$$

In practical implementations of the SPA, the following equivalent CN update rule is often used

$$L_{j \to i} = \left[\prod_{i' \in N(j) \setminus i} \text{sign}(L_{i' \to j}) \right] \cdot \phi^{-1} \left(\sum_{i' \in N(j) \setminus i} \phi(|L_{i' \to j}|) \right) \quad (7)$$

where $\phi(x) = -\log[\tanh(x/2)] = \log((e^x + 1)/(e^x - 1))$, $\phi^{-1}(x) = \phi(x)$, and $\phi(\infty) = 0$. In some fixed-point implementations, in order to have better approximation, different look-up tables could be used to compute $\phi(x)$ and $\phi^{-1}(x)$ [22].

We note that the hyperbolic tangent function, $\tanh(x)$, has numerical saturation problems when computed with finite

precision. For example, in 64-bit floating-point (in IEEE 754 standard format [28]) computer implementation, it can be shown that $\tanh(x/2)$ would be rounded to 1 when $x > 38$, meaning that $\phi^{-1}(\phi(x)) = \infty$ for $x > 38$ [29]. In order to avoid such problems that can arise from limited precision, thresholds on the magnitudes of messages must be applied in simulation studies [22].

In order to maintain the performance advantage of SPA decoding over MS decoding, the quantization method has to preserve the self-inverse property of the $\phi(x)$ function and to accurately compute the CN update function in (7). However, it is difficult to have a good approximation of the $\phi(x)$ function with limited resolution, because this requires both fine precision and large range. Efforts have been made to design quantization methods that work effectively with the $\phi(x)$ function. For example, a variable-precision quantization scheme proposed in [20] uses larger quantization step size for magnitudes greater than 1, and smaller step size for magnitudes less than 1. An adaptive uniform quantization method proposed in [21] uses different quantization step sizes for the outputs of the $\phi(x)$ and the $\phi^{-1}(x)$ function in (7). If the output of the $\phi(x)$ function is quantized with finite precision ϵ, inputs greater than $\phi^{-1}(\epsilon)$ can not be distinguished, and $\phi^{-1}(\epsilon)$ is quite small even for extremely fine precision, e.g., $\phi^{-1}(10^{-3}) \approx 7.6$ and $\phi^{-1}(10^{-6}) \approx 14.5$. Hence, the largest supported magnitude during decoding depends on the finest precision of quantization. This means that increasing the quantization range without improving the precision is not beneficial.

In order to avoid dealing with the $\phi(x)$ function, a variety of other CN update rules, most of which are approximations to the SPA, have been proposed. Some of these approximation are based on the following equivalent version of the SPA CN update rule represented by (6) or (7),

$$L_{j \to i} = \underset{i' \in N(j) \setminus i}{\boxplus} L_{i' \to j} \tag{8}$$

where $\boxplus$ is the pairwise "box-plus" operator defined as

$$x \boxplus y = \log \left(\frac{1 + e^{x+y}}{e^x + e^y} \right)$$
$$= \text{sign}(x)\text{sign}(y) \cdot \{\min(|x|, |y|) + s(|x|, |y|)\} \tag{9}$$
$$= \text{sign}(x)\text{sign}(y) \min(|x|, |y|) + s(x, y) \tag{10}$$

with

$$s(x, y) = \log \left(1 + e^{-|x+y|} \right) - \log \left(1 + e^{-|x-y|} \right). \tag{11}$$

The proof of equivalence between (6) and (8) can be found in [30]. We call such an implementation box-plus SPA decoding. The formulation above does not have the precision problem that (6) and (7) have, and, in fact, in 64-bit double-precision floating-point implementation, the maximum magnitude of a message that can be supported is approximately 1.8×10^{308}, which is the largest double-precision value supported by the IEEE 754 standard. Moreover, unlike the $\phi(x)$ function, the function $\log\left(1 + e^{-|x|}\right)$ can be well quantized or approximated with piecewise linear functions [29]–[31].

If the term $s(x, y)$ is omitted when using (8) to calculate the CN output in box-plus decoding, the result is the same as that produced by the MS algorithm using (3). Therefore, box-plus SPA decoding can be viewed as MS decoding with a correction factor. It is known that the magnitude of $s(x, y)$ is bounded above by $\log 2$ (see, for example, [33, p. 232]). In fact, as shown in [27], [32], given the same inputs, a message produced by a CN in SPA decoding has the same sign as the corresponding message in MS decoding, with equal or smaller magnitude. Because of their relevance to the proof of Theorem 4 below, we summarize these observations relating the CN updates produced by the SPA and MS decoders in Lemma 3.

Lemma 3: Let L_{MS} denote the message from CN j to VN i as computed in (3), and let L_{SPA} denote the message from CN j to VN i as computed in (6), (7), and (8). Then $\text{sign}(L_{SPA}) = \text{sign}(L_{MS})$ and $|L_{SPA}| \leq |L_{MS}|$. The correction term $s(x, y)$ in (11) satisfies $-\log 2 < s(x, y) < \log 2$, and $\text{sign}(s(x, y)) = -\text{sign}(xy)$ when $xy \neq 0$.

Proof: See Appendix C. ∎

Finally, we note that if the correction term $s(|x|, |y|)$ in (9) is replaced with a fixed constant, the resulting CN update rule corresponds to that of the OMS decoder in (5).

As we discussed earlier, no matter how one designs the fixed-point implementation of the original SPA using the $\phi(x)$ function, or even with the floating-point implementation, the function $\left| x - \phi^{-1}(\phi(x)) \right|$ is unbounded. Even if we saturate both the input and the output of the $\phi(x)$ function, the value of $\left| x - \phi^{-1}(\phi(x)) \right|$ is still unbounded and linear in x. Therefore, the CN output of a practical implementation of (6) or (7) can significantly differ from the true computed value. However, since box-plus SPA decoding can be considered as min-sun decoding with a correction factor, the implementation error mainly comes from the computation and quantization of the correction factor, which is a small bounded value, as shown in Lemma 3. Now, we can extend Theorem 1 to SPA decoding.

Theorem 4: Let G be the Tanner graph of a variable-regular LDPC code that contains a subgraph S induced by a trapping set. Assume that the channel is either a BSC or an AWGNC, and that the messages from the channel to all VNs outside S are correct. If S satisfies the k-separation assumption for sufficiently large k, then the SPA decoder will successfully correct all erroneous VNs in S.

Proof: See Appendix D. ∎

Remark 1: As will be shown in the simulation results, linear scaling of the input LLRs to the SPA decoder will indeed affect the decoding performance, because the correction factor $s(x, y)$ is not linear in either x or y.

For most LDPC codes, the trapping sets typically satisfy the k-separation assumption only for small values of k. Nevertheless, as described more fully in Section V, in our 64-bit double-precision floating-point computer simulations of MS decoding and box-plus SPA decoding applied to several LDPC codes traditionally associated with high error floors, we have not observed, in tens of billions of channel realizations of both the BSC and the AWGNC, any decoding failure in which the error patterns correspond to the support of a small trapping set. Moreover, when we force every VN in a trapping set to be in error and all other VNs to be correct, the floating-point decoders can successfully decode, whereas a decoder implementation that limits the magnitude of messages may

not be able to resolve the errors in the trapping set and would then fail to decode to the correct codeword.

We emphasize that the analytical and numerical results in this paper are mainly for variable-regular LDPC codes. Extension of this analysis to variable-irregular LDPC codes does not appear to be straightforward.

IV. NEW QUANTIZED DECODERS WITH LOW ERROR FLOORS

As mentioned above, several empirical studies have shown that the range and the precision of quantized messages in iterative LDPC decoders can influence the observed error floor. Moreover, analytical models used to study the dynamical evolution of messages show that message magnitudes can exhibit exponential growth behavior as a function of the number of decoder iterations. Likewise, the proofs of the theorems and corollaries in Section III suggest that iterative decoder performance can be improved by allowing for the exponential growth of message magnitudes. These results serve as the motivation for a new quantization method that we refer to as $(q + 1)$-*bit quasi-uniform* quantization, which we now describe.

Consider first the uniform quantizer with quantization step Δ. For any real number x, it is defined by

$$Q_\Delta(x) = sgn(x)\Delta \left\lfloor \frac{|x|}{\Delta} + \frac{1}{2} \right\rfloor.$$

The outputs of the uniform quantizer are of the form $m\Delta$. The quantization intervals can be visualized by expressing the quantization rule as

$$Q_\Delta(x) = \begin{cases} m\Delta, & \text{if } m\Delta - \frac{\Delta}{2} \leq x < m\Delta + \frac{\Delta}{2}, \\ & \text{for } m > 0 \\ 0, & \text{if } -\frac{\Delta}{2} < x < \frac{\Delta}{2} \\ m\Delta, & \text{if } m\Delta - \frac{\Delta}{2} < x \leq m\Delta + \frac{\Delta}{2}, \\ & \text{for } m < 0 \end{cases} \quad (12)$$

Now, let $N = 2^{q-1} - 1$, where q is an integer value $q \geq 1$. The q-bit uniform quantizer combines the uniform quantization intervals corresponding to the output values $m\Delta$, $\forall m \geq N$ into a single semi-infinite interval whose elements are quantized to $N\Delta$ and, similarly, combines the intervals corresponding to the output values $m\Delta$, $\forall m \leq -N$ into a single semi-infinite interval whose elements are quantized to $-N\Delta$. Denoting the q-bit quantizer with step Δ by $Q_{\Delta,q}(x)$, we have

$$Q_{\Delta,q}(x) = \begin{cases} N\Delta, & \text{if } N\Delta - \frac{\Delta}{2} \leq x \\ m\Delta, & \text{if } m\Delta - \frac{\Delta}{2} \leq x < m\Delta + \frac{\Delta}{2}, \\ & \text{for } N > m > 0 \\ 0, & \text{if } -\frac{\Delta}{2} < x < \frac{\Delta}{2} \\ m\Delta, & \text{if } m\Delta - \frac{\Delta}{2} < x \leq m\Delta + \frac{\Delta}{2}, \\ & \text{for } 0 > m > -N \\ -N\Delta, & \text{if } x \leq -N\Delta + \frac{\Delta}{2} \end{cases} \quad (13)$$

The number of intervals is $2N + 1 = 2^q - 1$, and the quantizer output levels $m\Delta$, $-N \leq m \leq N$, can be denoted by the signed q-bit binary representation of m, that is, $[m_0, m_1, \ldots, m_{q-1}]$, where the last $q - 1$ bits are the binary

representation of $|m|$, and m_0 is the sign bit with value 0 (resp. 1) when m is positive (resp. negative). Note that the output level 0 has two such binary representations; one of them can be selected using any preferred convention.

One approach to expanding the range of quantized messages is to increase the step size Δ, without changing the resolution q. This approach, however sacrifices the precision of the quantization. Alternatively, one could maintain the value of Δ and increase q to resolve larger magnitudes. This would increase implementation complexity when incorporated into the decoding hardware.

In the context of our application, the $(q + 1)$-*bit quasi-uniform* quantizer represents a compromise between these conflicting objectives of retaining fine precision, allowing large dynamic range, and controlling implementation complexity as messages grow exponentially in the number of decoder iterations. The definition of the quantizer involves another parameter $d > 1$, which we refer to as the *growth rate* parameter. Roughly speaking, the underlying idea behind the quantizer is as follows. For input values in the interval $(-dN\Delta, dN\Delta)$, we use q-bit uniform quantization with step size Δ. The intervals corresponding to quantized values $m\Delta$, $-(N - 1) \leq m \leq (N - 1)$ are exactly like those of the q-bit uniform quantizer. For values $N\Delta$ and $-N\Delta$, the semi-infinite intervals are shortened to have length $dN\Delta - N\Delta + \frac{\Delta}{2}$. For input values with magnitude larger than $dN\Delta$, the quantizer outputs can take an additional $N + 1 = 2^{q-1}$ values of the form $d^r N\Delta$, $1 \leq r \leq N+1$, with corresponding intervals that increase exponentially in length with growth rate d. More precisely, the $(q + 1)$-*bit quasi-uniform* quantizer, denoted by $Q^*_{\Delta,q}(x)$ is defined as follows.

$$Q^*_{\Delta,q}(x) = \begin{cases} d^{N+1}N\Delta, & \text{if } d^{N+1}N\Delta \leq x \\ d^r N\Delta, & \text{if } d^r N\Delta \leq x < d^{r+1}N\Delta, \\ & \text{for } N \geq r \geq 1 \\ Q_{\Delta,q}(x), & \text{if } -dN\Delta < x < dN\Delta \\ -d^r N\Delta, & \text{if } -d^{r+1}N\Delta < x \leq -d^r N\Delta, \\ & \text{for } 1 \leq r \leq N \\ -d^{N+1}N\Delta, & \text{if } x \leq -d^{N+1}N\Delta \end{cases} \quad (14)$$

From (14), we see that the quantization levels can be represented with only $q + 1$ bits. The levels $m\Delta$, $-N \leq m \leq N$ are represented by $[m_0, m_1, \ldots, m_{q-1}, m_q]$, where $[m_0, m_1, \ldots, m_{q-1}]$ is the signed binary representation of the integer m and the final *indicator bit*, m_q is set to zero, i.e., $m_q = 0$, to reflect the fact that the q-bit uniform quantizer has been applied. The 2^q quantized levels $d^r N\Delta$, $1 \leq r \leq N+1$ are denoted by $[r_0, r_1, \ldots, r_{q-1}, r_q]$, where $[r_0, r_1, \ldots, r_{q-1}]$ is the signed binary representation of $r - 1$, and the indicator bit r_q is set to 1, i.e., $r_q = 1$, to indicate that non-uniform quantization has been used. Similarly, we denote the 2^q quantized levels $-d^r N\Delta$, $1 \leq r \leq N + 1$ by $[r_0, r_1, \ldots, r_{q-1}, r_q]$, where $[r_0, r_1, \ldots, r_{q-1}]$ is the signed binary representation of $-(r - 1)$, and the indicator bit r_q is again set to 1, i.e., $r_q = 1$. It is sometimes convenient to represent these quantization levels in the form (y, b), where y is the decimal integer representation of the signed binary q-tuple $[m_0, m_1, \ldots, m_{q-1}]$ or $[r_0, r_1, \ldots, r_{q-1}]$, and b is the indicator bit m_q or r_q.

TABLE I
(3+1)-BIT QUASI-UNIFORM QUANTIZATION WITH $\Delta = 1$ AND $d = 3$.

Input range	Quantized value (decimal)	Binary form
[0,0.5]	0	0000
(0.5,1.5]	1	0010
(1.5,2.5]	2	0100
(2.5,9)	3	0110
[9,27)	9	0001
[27,81)	27	0011
[81,243)	81	0101
[243, ∞)	243	0111

TABLE II
4-BIT QUASI-UNIFORM QUANTIZATION WITH $\Delta = 1$, $d = 3$, AND $N_u = 5$.

Input range	Quantized value (decimal)	Binary form
(0,0.5]	0	0000
(0.5,1.5]	1	0001
(1.5,2.5]	2	0010
(2.5,3.5]	3	0011
(3.5,12)	4	0100
[12,36)	12	0101
[36,108)	36	0110
[108,∞)	108	0111

Table I shows an example of (3+1)-bit quasi-uniform quantization with $\Delta = 1$, $q = 3$, and $d = 3$. Here $N = 3$. The operation of the quantizer is shown only for non-negative real inputs. The operation on negative reals can be obtained by odd symmetry. The first bit is the sign bit, and the last bit is the indicator bit. The quantizer behaves just like the 3-bit uniform quantizer in the interval $[0, 9)$. When $x \geq 9$, the quantizer uses intervals of exponentially increasing length, with input x quantized to the smallest value in the interval in which x falls. For example, all values within the quantization interval $[27, 81)$ are quantized to 27. The decimal values are used in the VN and CN update computations, and then the corresponding quantized binary messages are passed between VNs and CNs.

We can further extend the idea of (q+1)-bit quasi-uniform quantization, as follows. The $(q + 1)$-bit quasi-uniform quantizer uses $q + 1$ bits in total to represent $2N + 1 = 2^q$ different output magnitudes, or $2^{q+1} - 1$ quantization intervals if signs are taken into account. As described in (14) and illustrated in Table I, $N + 1 = 2^{q-1}$ output magnitudes, including 0, are allocated to the uniform quantization domain and the remaining $N + 1 = 2^{q-1}$ magnitudes correspond to exponentially growing quantization interval lengths. The generalized (symmetric) $(q + 1)$-bit quasi-uniform quantizer represents the same number of magnitudes, but it can assign any number, say N_u, to the uniform quantization range and the remaining $2^q - N_u$ magnitudes to the exponential quantization range. With a quantization rule similar to (14), the quantized values of the general $q + 1$-bit quasi-uniform quantization are $m\Delta$ for $-N_u < m < N_u$; $d^{r-N_u+1}N_u\Delta$ for $r \geq N_u$, and $-d^{r-N_u+1}N_u\Delta$ for $r \leq -N_u$. Table II shows an example of a general 4-bit quasi-uniform quantization with $N_u = 5$, $\Delta = 1$, and $d = 3$. The uniform quantization range in this example is from -4 to 4 with uniform step size 1, and the exponential range is above 4 or below -4 with exponential step sizes $4 \cdot (3^r - 3^{r-1})$ for $1 \leq r \leq 3$.

The motivation for the proposed quasi-uniform quantization method was the analysis of message-passing decoder behavior on trapping sets that satisfy the k-separation assumption for large k. Although this property is generally not satisfied by trapping sets in practical LDPC codes, the simulation results in the next section demonstrate that, for a variety of LDPC codes that were examined, this quantization approach can substantially lower error floors when used with standard MS-based and SPA-based decoders.

V. NUMERICAL RESULTS

In this section we compare the error-rate performance obtained with the proposed quasi-uniform quantization method to that obtained using uniform quantization. We consider four known LPDC codes covering a range of rates and lengths: a rate-$\frac{3}{10}$, (640,192) quasi-cyclic (QC) LDPC code [17]; the rate-$\frac{1}{2}$, (2640,1320) Margulis LDPC code [4]; the rate-$\frac{4}{5}$, (1280, 1024) AR4JA LDPC code [36]; and MacKay's (4095,3358) regular LDPC code (the 4095.737.3.101 code in [35]) with rate approximately 0.82. Results are shown for various combinations of the BSC and AWGN channels using the MS, OMS, AMS, SPA, and approximated-SPA decoders.

All of the frame error rate (FER) curves are based on Monte Carlo simulations that generated at least 200 error frames for each plotted error rate, and the maximum number of decoding iterations was set to 200, unless otherwise indicated.

A. Dynamical Range of Message Magnitudes

We first present some empirical data in support of the contention that some benefit may come from allowing message magnitudes to grow during iterative decoding.

Fig. 2 shows the empirical probability density functions (pdf) of the message magnitudes observed during decoding simulations for two LDPC codes. Fig. 2(a) shows the pdf for the MS decoder applied to the (640,192) QC-LDPC code on the BSC with $p = 0.03$, where the magnitude of all input LLRs is scaled to 1. Fig. 2(b) shows the pdf of the SPA decoder applied to the Margulis code on the AWGNC with $E_b/N_0 = 2.25$ dB. The data used to create these figures were obtained using floating-point decoder implementations and more than 10 million channel output symbols. The messages passed on all edges during all decoding iterations were collected to generate the pdfs. In the simulations, the iterative MP decoders stopped when a codeword was found or when the maximum number of iterations (200) was reached. The figures confirm that a substantial fraction of messages had "large" magnitudes. Moreover, upon further examination of the simulation data, we found that such "strong" messages, in general, helped to successfully decode the received symbols, as suggested by the idealized theoretical analysis in Section III.

B. Simulation Results for Min-Sum Decoding and Variants

Figs. 3 and 4 show simulation results for the (640,192) QC-LDPC code using various types of quantized MS decoders and floating-point MS decoders, extending some of the results presented in [37]. For the BSC, we scaled the magnitudes of decoder input messages from the channel to 1 since, for linear
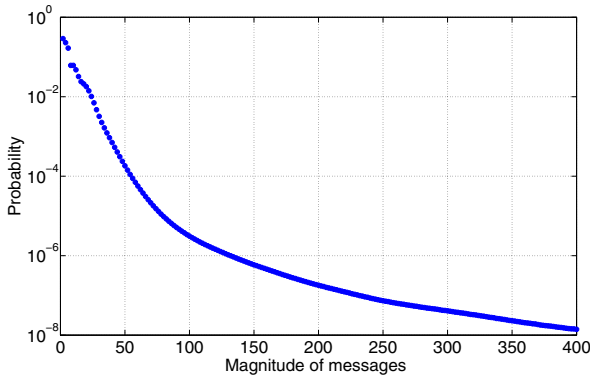
(a) MS decoder on the (640,192) QC-LDPC code on BSC of $p = 0.03$ and $|LLR| = 1$.
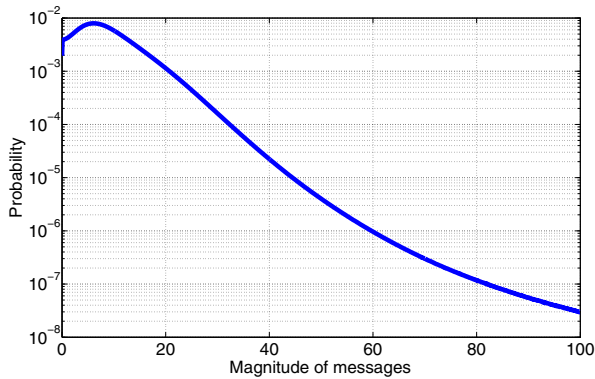
(b) SPA decoder on the Margulis code of length 2640 on AWGNC of $E_b/N_0 = 2.25$ dB.

Fig. 2.   Probability density function of magnitude of messages.

Fig. 3.   FER results of min-sum (MS) decoder on the (640,192) QC-LDPC code on BSC, where $\Delta = 1$ or 0.5, and $d = 3$.

Fig. 4.   FER results of min-sum (MS) decoder on the (640,192) QC-LDPC code on AWGNC, where $\Delta = 0.5$ and $d = 3$.

decoders such as Gallager-B and MS, the scaling of channel input messages does not affect the decoding performance. The uniform quantization step size Δ is set to 1 or 0.5. So, for example, when $\Delta = 1$, the 3-bit uniform quantizer produces values $\{\pm 3, \pm 2, \pm 1, 0\}$, and the (3+1)-bit quasi-uniform quantizer with $d = 3$ yields the values described in Table I. Notice that setting step size to 0.5, which is half of the input LLR magnitude, is equivalent to having $\Delta = 1$ while scaling the input LLR magnitude to 2. In the simulation, the parameter d was heuristically chosen by testing different values. When q is large, a small d proved to be enough to represent a large range of message magnitudes.

In Fig. 3, we see that the slope of the error floor resulting from uniform quantization with either step size, $\Delta = 1$ or $\Delta = 0.5$, is similar to that of the Gallager-B decoder error floor. This is because, when most messages have the same magnitude, MS decoding essentially degenerates to Gallager-B decoding, which relies solely upon the signs of messages.

Comparing uniform quantizers with the same number of bits but different step sizes, we see that smaller step size produces better performance in the waterfall region but a higher error floor. This observation can be explained by the saturation level of these quantizers. For example, 3-bit and 4-bit uniform quantizers with step size $\Delta = 1$ saturate at magnitudes 3 and 7, respectively, whereas with step size $\Delta = 0.5$, they saturate at magnitudes 1.5 and 3.5, respectively. The stronger messages, i.e., the messages with larger magnitudes, can be helpful
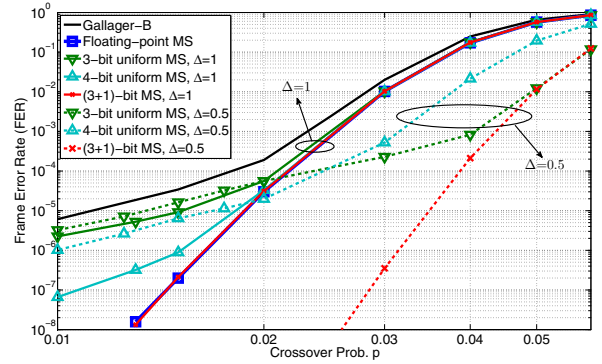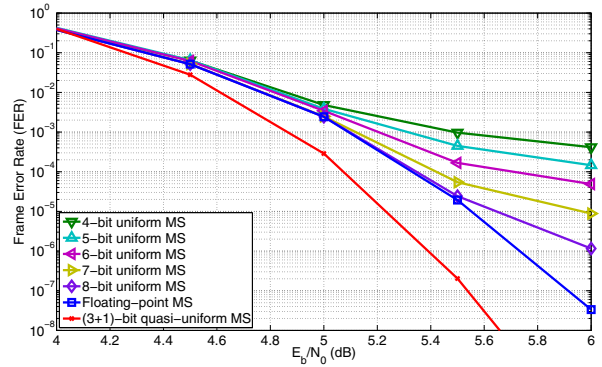
or harmful to the decoding process, depending on whether they are correct or not. The correct ones can help overcome the incorrectly received bits, but the incorrect ones tend to negatively influence the recovery of correctly received bits. In the error-floor region, when channel conditions are good, very few bits are received incorrectly, and as suggested by the proofs of Theorems 1 and 4, large saturation levels allow messages corresponding to correct bits to grow sufficiently to overcome the "incorrect" messages in trapping sets. This behavior is evident in Fig. 3, where the error floors produced by the different uniform quantizers monotonically decrease as the saturation levels increase.

On the other hand, in the waterfall region where many bits are received incorrectly, reducing the saturation level may limit the propagation of strong incorrect messages. Moreover, in this specific case, quantization with the smaller step size $\Delta = 0.5$ may be expected to improve performance relative to that achieved with the larger step size $\Delta = 1$ or with a floating-point MS decoder implementation. The reasoning is that, since the magnitudes of input LLRs to the MS decoder from the BSC are scaled to 1, the low saturation level and the possible appearance of non-integral saturated messages may reduce the possibility of the messages at a VN summing to zero. Because having VNs summing to zero could result in oscillatory behavior in the decoder and failure to decode correctly, this could explain why in Fig. 3 the MS decoder using (3+1)-bit quasi-uniform quantization and step size $\Delta = 0.5$ yields better
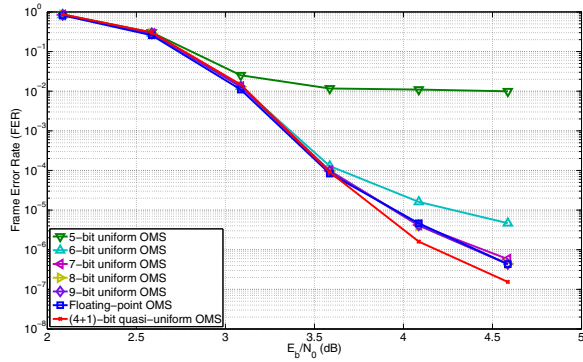
Fig. 5. FER results of OMS decoder on the AR4JA LDPC code of $k = 1024$ and $r = 0.8$ on AWGNC, where $\Delta = 0.5$, $d = 1.5$, and offset factor $\beta = 0.5$,.
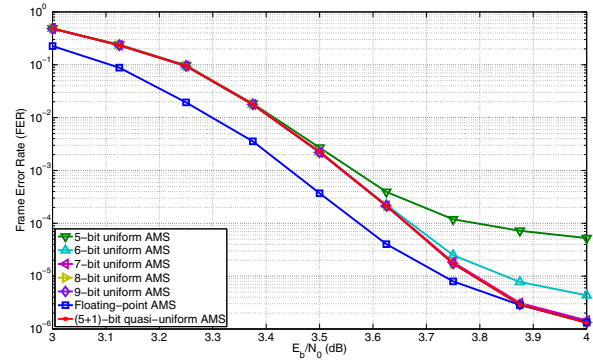

Fig. 6. FER results of AMS decoder on the (4095,3358) LDPC code on AWGNC, where $\Delta = 0.5$, $d = 1.3$, and attenuation factor $\alpha = 0.7$.

performance than the floating-point decoder.

Fig. 4 shows the performance of MS decoding of the (640,192) QC-LDPC code on the AWGNC channel. Here the $(3+1)$-bit quasi-uniform quantizer yields substantial reduction of the error floor in comparison not only to 8-bit uniform quantization but also to the floating-point results. This is consistent with, and more impressive than, the results shown in [37] for the Margulis code, where $(5+1)$-bit quasi-uniform quantization surpassed 6-bit uniform quantization and paralleled floating-point results. Heuristic reasoning along the lines used above suggests that codes with higher variable-node degree would benefit even more from the quasi-uniform quantization. However, it is important to point out that the gains can be code-dependent, so further performance studies are needed to confirm this.

Quasi-uniform quantization can be directly applied to modi-fied MS decoders, such as AMS and OMS, with the possibility of significant reduction in the error floor. This was illustrated in [37] for the (640,192) QC-LDPC code with AMS decoding on the BSC and with OMS decoding on the AWGNC. In case of AMS decoding, $(3+1)$-bit quasi-uniform quantization dramatically reduced the error floor relative to 4-bit uniform quantization, achieving the performance of the unsaturated AMS decoder. For OMS decoding with $(4+1)$-bit quan-tization, the comparisons to 5-bit uniform quantization and unsaturated decoding were analogous.

Here we consider the performance of AMS and OMS de-coding on longer codes with higher rates, specifically, the rate-0.8, (4095,3358) regular code and the rate-0.8, (1280, 1024), irregular AR4JA code. Fig. 5 compares the quasi-uniform quantization method with uniform quantization in OMS de-coding. The performance of the floating-point OMS decoder is also shown. With uniform quantization ranging from 5 bits to 9 bits, we can see that 8 bits suffice to closely approach the error-rate performance of floating-point OMS, whereas the $(4+1)$-bit quasi-uniform quantization actually surpasses floating-point decoder. Fig. 6 shows a similar comparison for AMS decoding of MacKay's (4095,3358) LDPC code. The attenuation factor α was set to the value 0.7, which was found empirically to give the best error floor performance among integer multiples of 0.1 in the range [0.5, 0.9]. After normalization by this factor in every CN update, we found that

the quantized value lost precision due to the coarse step size $\Delta = 0.5$. As a consequence, the floating-point AMS decoder had better performance than any of the quantized decoders, most noticeably in the waterfall region. Uniform quantization with 7 or more bits appears to eventually achieve floating point performance at FER below 3×10^{-6}, as does $(5+1)$-bit quasi-uniform quantization.

C. Simulation Results for Sum-Product Algorithm Decoding

We now consider the application of quasi-uniform quanti-zation to SPA decoding. In our simulations of quantized SPA decoding, the input LLRs and the messages passed between CNs and VNs are quantized values. For convenience, the CN updates are carried out with floating-point arithmetic using the box-plus update rule in (8); the resulting message is then quantized appropriately.

In [38], we illustrated the performance of quasi-uniform quantization with SPA decoding of the (640,192) QC-LDPC code on the BSC. We saw that with LLR magnitudes scaled to 2, the $(6+1)$-bit quasi-uniform quantizer with step size $\Delta = 0.25$ and $d = 1.5$ performs significantly better than 7-bit uniform quantization with the same step size. Its performance is comparable to that of the floating-point SPA decoder, which is superior to floating-point SPA decoding with exact LLR magnitudes $\log \frac{1-p}{p}$ when the channel error probability p is small.

Here we consider the same code and channel, with step size again set to $\Delta = 0.25$, but with quantization value scale factor reduced to $d = 1.3$. With LLR magnitudes scaled to 2, we simulated 6-bit through 10-bit uniform quantization, $(5+1)$-bit quasi-uniform quantization, and floating-point SPA decoding with LLR magnitudes scaled to 2 as well as with exact LLR magnitudes.

The simulation results, shown in Fig. 7, indicate that the $(5 + 1)$-bit quasi-uniform quantizer provides the best perfor-mance for $p < 0.06$. Comparing to the results in [38], the performance of the $(5 + 1)$-bit quantizer with $d = 1.3$ is only slightly worse than that of the $(6 + 1)$-bit quantizer with $d = 1.5$.

We note that the selection of the input LLR magnitude, here set to 2, is heuristic and code-dependent. The value 2 was found empirically to give much better performance than,
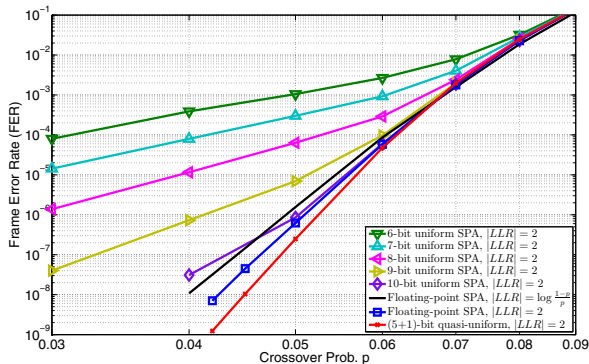
Fig. 7. FER results of SPA decoder on the (640,192) QC-LDPC code on BSC, where $\Delta = 0.25$, and $d = 1.3$.

Fig. 8. FER results of approx.-SPA decoder on the (2640,1320) Margulis code on AWGNC, where $\Delta = 0.25$ and $d = 1.3$.

for example, the value 1, but does not necessarily represent the optimal LLR magnitude scaling.

Results for SPA decoding of the (640,192) QC-LDPC code on the AWGN channel were also presented in [38]. The $(6 + 1)$-bit quasi-uniform quantizer with $\Delta = 0.25$ and $d = 1.5$ was found to significantly improve upon 7-bit uniform quantization and match the performance of the floating-point box-plus SPA decoder.

In [38], we found similar relative performance for the Margulis code on the AWGNC. The $(6+1)$-bit quasi-uniform quantizer outperformed 7-bit uniform quantization, with step size parameters $\Delta = 0.25$ and $d = 1.2$, and its performance equaled that of the "approximated box-plus SPA" decoder. The latter made use of a two-piece linear approximation for $\ln(1 + e^{-|x|})$, taken from [31], in computing the correction factor $s(x, y)$ for box-plus SPA decoding in (11), namely,

$$\ln\left(1 + e^{-|x|}\right) = \begin{cases} 0.6 - 0.24|x|, & \text{if } |x| < 2.5 \\ 0, & \text{otherwise.} \end{cases} \quad (15)$$

The approximated decoder ran about five times faster than the floating-point SPA decoder, with a performance penalty of less than 0.02 dB in the waterfall region.

In Fig. 8 we show further results for the Margulis code on the AWGNC. The plot shows the FER results for $(5 + 1)$-bit quasi-uniform quantization, as well as 6-bit, 7-bit, and 8-bit uniform quantizers, with quantization parameters set to $\Delta = 0.25$ and $d = 1.3$. We also evaluated the dual quantization SPA decoding proposed by Zhang *et al.* [21], where the $\phi(x)$ function is quantized into a mapping table, denoted as $\bar{\phi}(x)$. Following the notation in [21], we considered dual quantization with parameters Q4.2/1.5, Q5.2/1.6, and Q6.2/1.7 for 6-bit, 7-bit, and 8-bit quantizers, respectively. The Q$m.f$ quantizer uses uniform quantization to represent a signed fixed-point number with m bits to the left of the radix point for the integer part and f bits to the right of the radix point for the fractional part. For example, a Q4.2 quantizer has uniform quantization step size of 0.25 and a range $[-7.75, 7.75]$. Hence, all the quantization methods compared here have the same uniform step size of $\Delta = 0.25$ when quantizing the input LLRs.

We know that the saturation level $\bar{\phi}(0)$ is limited by the quantization step size, because it is desirable to have $\bar{\phi}(0) < x$ for all x satisfying $\bar{\phi}(x) = 0$. In other words, in the dual
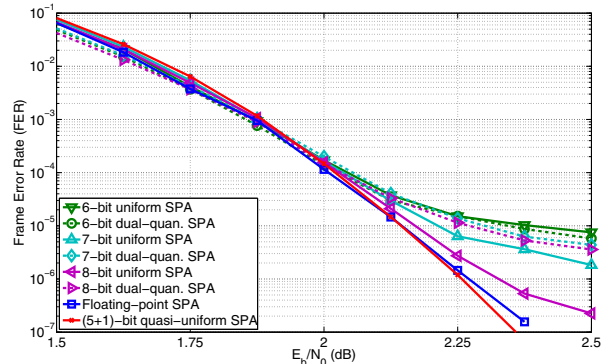
quantization scheme, the saturation level has to match the resolution of the quantizer; otherwise the error-rate performance in both the waterfall region and the error-floor region will be significantly degraded. Based on error-rate simulations using a range of saturation levels for dual quantization methods, we chose the saturation level for $\bar{\phi}(x) = 0$ to be 5.5, 7, and 8 for the 6-bit, 7-bit, and 8-bit dual quantizers, respectively. As the figure reveals, the $(5 + 1)$-bit quasi-uniform quantizer yields the best FER performance in the error-floor region.

We also evaluated the performance of quasi-uniform quantization in the context of decoding an irregular LDPC code, namely the rate-$\frac{4}{5}$, $(1280, 1024)$ AR4JA code. This protograph-based code has variable node degrees ranging from 1 to 6. Fig. 9 shows the FER obtained with approximated-SPA decoding and (5+1)-bit quasi-uniform quantization, with $\Delta = 0.5$ and $d = 1.3$. Also shown are the results obtained with the floating-point decoder, as well as those produced by 8-bit uniform quantization with step size $\Delta = 0.125$. The $(5+1)$-bit scheme was superior to both of these alternatives. The figure also includes two curves taken from [36], corresponding to an 8-bit quantized SPA decoder with modified VN update rules that were designed specifically for this code, as well as a "fully-optimized" 8-bit decoder with more sophisticated VN/CN update rules. The $(5+1)$-bit quasi-uniform quantizer's performance surpassed that of the former, but it could not match that of the fully-optimized 8-bit decoder.

D. Effect of Iteration Limits

Figs. 4 – 9 show that $(q+1)$-bit quasi-uniform quantization can provide attractive error-floor performance, sometimes even better than the double-precision floating-point box-plus SPA decoder. In generating these results, we observed from the simulation data that the floating-point SPA generally requires more iterations to decode a codeword than the quasi-uniform quantized SPA, especially in the high SNR region. Since the maximum number of iterations was set to 200 in our simulations, the faster convergence of the quasi-uniform quantized SPA allowed it to outperform the floating-point SPA scheme. The convergence properties of the quasi-uniform quantized SPA decoder appear to derive from its use of non-uniform, exponentially growing step sizes. From the theoretical analysis discussed in Section III, we know that the exponential growth
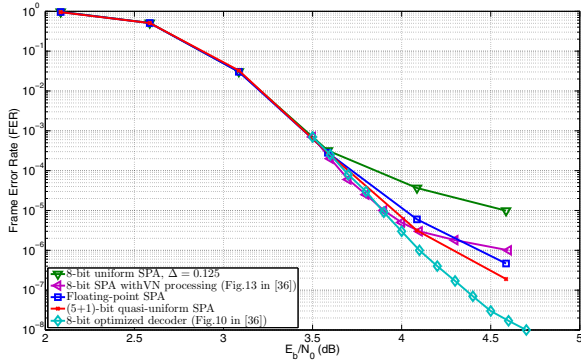
Fig. 9. FER results of approx.-SPA decoder on the AR4JA LDPC code of $k = 1024$ and $r = 0.8$ on AWGNC, where $\Delta = 0.5$ and $d = 1.3$.

Fig. 10. FER results of approx.-SPA decoder on the (4095,3358) LDPC code on AWGNC, where $\Delta = 0.5$ and $d = 1.3$.

rate of correct messages is larger than that of incorrect messages. We might expect that, with a properly designed quasi-uniform quantizer, the correct messages can reach the higher magnitude level earlier than the incorrect messages, and therefore incorrect messages are more likely to be quantized to lower magnitude levels. Hence, the correct messages can "overcome" the incorrect messages more rapidly, allowing the decoder to converge to a codeword after fewer iterations.

In Fig. 10, we explore the effect of limiting the number of iterations in approximated-SPA decoding of MacKay's rate-0.82, (4095,3358) LDPC code. With the maximum number of iterations set to 200, we show the results for 6-bit and 10-bit uniform quantizers, the $(5 + 1)$-bit quasi-uniform quantizer, and the floating-point decoder. We also compare the performance of $(5 + 1)$-bit quasi-uniform quantization and the floating-point decoder when the maximum number of iterations is raised to 10K and even further to 100K.

With a limit of 200 iterations, this code manifested a high error floor with floating-point SPA decoding. The error floor was lower when the number of iterations could go as high as 10K, and dropped even further when up to 100K iterations were allowed. However, even in the latter case, the FER was only slightly lower than that found with the quasi-uniform quantizer with no more than 200 iterations. The performance of the quasi-uniform quantizer continued to improve in raising the limit to 10K and then to 100K. These results seem to be consistent with the intuition suggested by the theoretical analysis.

VI. CONCLUSION

Trapping sets and other error-prone substructures are known to influence the error-rate performance of LDPC codes with iterative message-passing decoding. In this paper, we have shown that the use of uniform quantization in iterative MP decoding can be a significant factor contributing to the error floor phenomenon in LDPC code performance. An analysis of iterative MP decoding in an idealized setting suggests that decoder message saturation plays a key role in the occurrence of errors in small trapping sets, leading to observed error floor behaviors. To address this problem, we proposed a novel quasi-uniform quantization method that effectively extends the dynamic range of the quantizer. Without modifying the CN and VN update rules or adding extra stages to standard
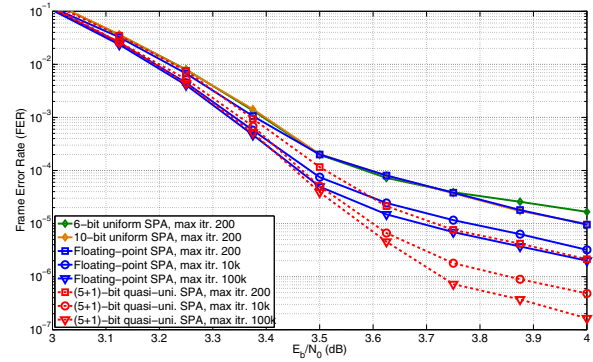
iterative decoding algorithms, the use of this quantizer was shown to significantly lower the error floors of several well-studied LDPC codes when used with various iterative MP decoding algorithms on the BSC and AWGNC. Simulation results confirmed that this new quantization method can significantly reduce the error floors of these codes with essentially no increase in decoding complexity.

APPENDIX A
PROOF OF THEOREM 1

Proof: Assume VN $v_r \in V_1 \subseteq S$ is k-separated and the corresponding computation tree is $T(v_r)$. Let $c_r \in C_1$ be the neighboring degree-one CN of v_r in S. From the separation assumption and the assumed correctness of channel messages for VNs outside S, all descendants of c_r in $T(v_r)$ receive correct initial messages from the BSC. Like the LLRs of the BSC outputs, all the initial messages in the decoder, L_i^{ch}, $1 \le i \le n$, have the same magnitude. Denote the subtree starting with CN c_r as $T(c_r)$. With the VN/CN update rules of the MS decoder, we analyze the messages sent from the descendants of c_r in $T(c_r)$. First, according to the CN update rule described in (3), all messages received by a VN from its children CNs in $T(c_r)$ must have the same sign as the message received from the channel by this VN, because all the messages passed in $T(c_r)$ are correct. Therefore, the outgoing message from any VN v_i to its parent CN c_j in $T(c_r)$ satisfies the following equality

$$|L_{i \to j}| = \left| L_i^{ch} + \sum_{j' \in N(i) \backslash j} L_{j' \to i} \right| = \left| L_i^{ch} \right| + \sum_{j' \in N(i) \backslash j} |L_{j' \to i}|. \tag{16}$$

Moreover, since the LDPC code considered is variable-regular and all the channel messages from the BSC have the same magnitude, all incoming messages received by a VN from its children CNs in $T(c_r)$ must have the same magnitude as well. Therefore, all the messages sent from VNs in the same level of the computation tree $T(c_r)$ have the same magnitude. Let $|L_l|$ be the magnitude of the messages sent by the VNs whose shortest path to a leaf VN contains l CNs in $T(c_r)$; in particular, $|L_0|$ is the magnitude of messages sent by leaf VNs, as well as the magnitude of channel inputs. The discussion

above implies that

$$|L_l| = |L_0| + (d_v - 1)|L_{l-1}| > (d_v - 1)|L_{l-1}| > (d_v - 1)^l |L_0|, \quad (17)$$

where d_v is the variable node degree. Hence, it can be seen that the magnitudes of messages sent towards the root CN c_r of the computation tree $T(c_r)$ grow exponentially, with $d_v - 1$ as the base, in every upper VN level. Therefore, for $l \le k$, the magnitude of the message sent in the l-th iteration from c_r to its parent node v_r, the k-separated root VN of $T(v_r)$, is greater than $(d_v - 1)^l |L_0|$.

Now, let us look at the subtree of $T(v_r)$ that has as its root a child CN $c' \in C_S \setminus C_1$ of the root v_r. Denote this subtree by $T(c')$. We assume that the message L'_l received by v_r from c' after l iterations has a different sign than the message received from $c_r \in C_1$; otherwise, v_r would already have been corrected. Now consider any subtree of $T(c')$ that has as its root a VN $v \in S$ and contains t levels of VNs. We denote such a tree by $T^t(v)$. If $t \ge 2a$, the subtree $T^t(v)$ must include at least one CN from the set C_1. To see this, recall that the induced subgraph of the trapping set is connected. Since there are a VNs in the trapping set, it follows that any two VNs in the trapping set can be connected by a path of length less than $2a$. Therefore, for $t \ge 2a$, $T^t(v)$ actually includes all the CNs and VNs in the induced subgraph of the trapping set, in particular a CN from C_1. Of course, for most trapping sets, $T^t(v)$ can include a CN from C_1 with t much smaller than $2a$.

Now, consider $T^t(v)$ as a "super-node" with $(d_v - 1)^t$ children VNs. Since $T^t(v)$ includes a CN from C_1, at least one of these children VNs has the property that all of its descendants receive correct messages from the channel. This means that at least one of the incorrect messages going into the super-node would be canceled out by one or more such correct messages. So if the output message, L_{out}, of such a super-node is incorrect, its magnitude satisfies

$$|L_{out}| < ((d_v - 1)^t - 1)|L_{in}| + |\bar{L}_{ch}|, \quad (18)$$

where $|L_{in}|$ is the largest magnitude of all incoming incorrect messages, and the second term $|\bar{L}_{ch}| \triangleq |L_0| \sum_{i=0}^{t-1} (d_v - 1)^i$ is an upper bound on the sum of the channel input LLRs to all of the VNs in the t-level subtree. Note that the leaf VNs of $T^t(v)$ are not necessarily the leaf VNs of $T(v_r)$. Thus, we can upper bound the magnitude of the incorrect message sent from c' to v_r after l iterations by

$$
\begin{aligned}
|L'_l| &< |L_0| \cdot [(d_v - 1)^t - 1]^{\lceil l/t \rceil} \\
&\quad + |\bar{L}_{ch}| \sum_{i=0}^{\lceil l/t \rceil - 1} [(d_v - 1)^t - 1]^i \quad (19) \\
&< (|L_0| + |\bar{L}_{ch}|) \cdot [(d_v - 1)^t - 1]^{\lceil l/t \rceil}
\end{aligned}
$$

where $\lceil x \rceil$ is the smallest integer greater than or equal to x. The upper bound in (19) is extremely loose, and for most small-size trapping sets, the upper bound is generally less than $|L_0|(d_v - 2)^l$.

Therefore, by taking the logarithms of $|L_l|$ in (17) and $|L'_l|$ in (19), respectively, we have

$$
\begin{aligned}
\log |L_l| &> \log |L_0| + l \log(d_v - 1) \\
&= \log |L_0| + l \cdot \frac{1}{t} \cdot \log(d_v - 1)^t, \quad (20)
\end{aligned}
$$

and

$$
\begin{aligned}
\log |L'_l| &< \log(|L_0| + |\bar{L}_{ch}|) + \lceil l/t \rceil \log[(d_v - 1)^t - 1] \\
&< \log(|L_0| + |\bar{L}_{ch}|) + \log[(d_v - 1)^t - 1] \\
&\quad + l \cdot \frac{1}{t} \cdot \log[(d_v - 1)^t - 1]. \quad (21)
\end{aligned}
$$

Note that the first term in (20) and the first two terms in (21) are constants and independent of the number of iterations l.

Since $\log(d_v - 1)^t > \log[(d_v - 1)^t - 1]$, if l is large enough and there is no limitation imposed on the magnitude of messages, it is easy to see from (20) and (21) that $|L_l|$ would be greater than $|L'_l|$ multiplied by any constant. This means that the correct messages coming from outside of the trapping set to VNs in V_1 through their neighboring CNs in C_1 will eventually have greater magnitude than the sum of incorrect messages from other neighboring CNs, i.e., $|L_l| > (d_v - 1)|L'_l|$. Hence, all the erroneous VNs in V_1 will be corrected. Since, by definition, an absolute trapping set does not contain a stopping set, the remaining erroneous VNs must form a smaller absolute trapping set. Therefore, we can use the same argument to show that as the number of iterations continues to grow, the correct messages would eventually be large enough to correct all erroneous VNs.

Now, we show that the proof technique above can be extended to the AWGNC. Define $|L_{\min}|$ and $|L_{\max}|$ to be the minimum and maximum magnitudes, respectively, of the input LLRs from the AWGNC. In this setting, the bounds on $\log |L_l|$ and $\log |L'_l|$ corresponding to those in (20) and (21) take the form

$$\log |L_l| > \log |L_{\min}| + l \cdot \frac{1}{t} \cdot \log(d_v - 1)^t, \quad (22)$$

and

$$
\begin{aligned}
\log |L'_l| &< \log(|L_{\max}| + |\bar{L}_{ch}|) + \log[(d_v - 1)^t - 1] \\
&\quad + l \cdot \frac{1}{t} \cdot \log[(d_v - 1)^t - 1]. \quad (23)
\end{aligned}
$$

Since the quantities $\log |L_{\min}|$ and $\log |L_{\max}|$ are constant and do not change as l increases, we can conclude, as we did for the BSC, that the correct messages from outside the trapping set will eventually have greater magnitude than the incorrect messages from within the trapping set. Therefore, all of the VNs will eventually be correctly decoded. ∎

APPENDIX B
PROOF OF COROLLARY 2

Proof: We first consider AMS decoding. Referring to the proof of Theorem 1 for the BSC case, we can replace the quantity $(d_v - 1)$ in (20) and (21) by $\alpha(d_v - 1)$, where α is the attenuation factor. In practice, we would always choose α such that $\alpha(d_v - 1)$ is greater than 1; otherwise, the error-correction performance of the AMS decoder would be inferior to that of the MS decoder. Similar reasoning to that used in the proof of the theorem then leads to the desired conclusion. For the AWGNC case, we make the corresponding changes in (22) and (23), and argue similarly.

For the OMS decoder, the proof follows from the proof of Theorem 4 in Appendix D. There, we simply replace the quantity $\bar{s}$ by the offset β. ∎

APPENDIX C
PROOF OF LEMMA 3

Proof: The first statement regarding the relationship between the sign and magnitude of the CN messages L_{SPA} and L_{MS} is proved in [32], [27]. For completeness, we include here an elementary alternative proof.

First note that if $xy = 0$, then $s(x, y) = 0$. Now, if x and y are nonzero and have the same sign (i.e., $xy > 0$), then $|x + y| > |x - y|$ and hence $s(x, y) < 0$. Hence, we can see from (9) that the first statement is true if the inequality $\min(x, y) + s(x, y) > 0$ holds for any positive real values x and y. Without loss of generality, if we assume $x \geq y > 0$, then the following inequalities are equivalent

$$
\begin{aligned}
& \min(x, y) + s(x, y) > 0 \\
\Leftrightarrow \quad & \log e^y + \log \frac{1 + e^{-x+y}}{1 + e^{-x-y}} > 0 \\
\Leftrightarrow \quad & e^y + e^{-x} - 1 - e^{-x+y} > 0 \\
\Leftrightarrow \quad & (e^y - 1)(1 - e^{-x}) > 0.
\end{aligned}
$$

Since $e^y > 1$ and $e^{-x} < 1$, the final inequality holds. Hence, the first statement is proved.

To prove the second statement, note that

$$
\begin{aligned}
s(x, y) & = \log \frac{1 + e^{-x-y}}{1 + e^{-x+y}} \\
& = \log \frac{e^x + e^{-y}}{e^x + e^y} \\
& \geq \log \frac{e^x + e^{-x}}{e^x + e^x} \\
& > \log \frac{1}{2} = -\log 2.
\end{aligned}
$$

Therefore, $-\log 2 < s(x, y) < 0$. When $xy < 0$, a similar line of reasoning shows that $s(x, y) > 0$ and $0 < s(x, y) < \log 2$. ∎

APPENDIX D
PROOF OF THEOREM 4

Proof: From Lemma 3, we know that a CN message in SPA decoding has the same sign as the corresponding CN message in MS decoding. Moreover, the magnitude of the former is less than or equal to that of the latter. To compute the output for a CN of degree d_c, the box-plus SPA uses the pairwise box-plus operation (10) at most $\log(d_c - 1)$ times. Hence, the difference between output messages of the SPA and the MS algorithm is upper bounded by $\bar{s} \triangleq \lceil \log(d_c - 1) \rceil \cdot \log 2$.

By applying an approach similar to that used in the proof of Theorem 1, we can lower bound the magnitude of messages L_l in SPA decoding as follows

$$
\begin{aligned}
|L_l| & > |L_0| + (d_v - 1)(|L_{l-1}| - \bar{s}) \\
& > (d_v - 1)^l |L_0| - \bar{s} \sum_{i=1}^{l} (d_v - 1)^i \\
& = (d_v - 1)^l \left(|L_0| - \frac{d_v - 1}{d_v - 2} \bar{s} \right) + \bar{s} \frac{d_v - 1}{d_v - 2}.
\end{aligned}
$$

Since all input messages to the decoder from the BSC have the same magnitude, if we scale the magnitudes of all initial messages such that

$$
|L_0| > \frac{d_v - 1}{d_v - 2} \bar{s} = \frac{d_v - 1}{d_v - 2} \cdot \lceil \log(d_c - 1) \rceil \cdot \log 2, \quad (24)
$$

then the magnitudes of messages sent towards c_r in the computation tree $T(v_r)$ grow exponentially in the number of

iterations, with base $d_v - 1$. Hence, using the same reasoning as in the proof of Theorem 1, it can be shown that, if k is large enough and there is no limit on the magnitudes of messages, the correct messages from outside the trapping set eventually overcome the incorrect messages passed within the trapping set, thereby correcting all erroneous VNs in the trapping set.

The extension to the AWGNC case is analogous to that used in Theorem 1. Let $|L_0|$ denote the minimum magnitude of all input LLRs from the AWGNC, and linearly scale the magnitudes of all the input messages such that the inequality (24) is satisfied. Then, reasoning as in the proof of the BSC case above, we can show that the magnitudes of correct messages outside the trapping set still grow exponentially with $d_v - 1$ as the base, and eventually they correct all erroneous VNs in the trapping set. ∎

REFERENCES

[1] R. G. Gallager, "Low-density parity-check codes," *IRE Trans. Inf. Theory*, vol. 8, pp. 21–28, Jan. 1962.

[2] D. J. MacKay and R. M. Neal, "Near Shannon-limit performance of low-density parity check codes," *Electron. Lett.*, vol. 33, pp. 457–458, Mar. 1997.

[3] C. Di, D. Proietti, I. E. Telatar, T. Richardson, and R. Urbanke, "Finite-length analysis of low-density parity-check codes on the binary erasure channel," *IEEE Trans. Inf. Theory*, vol. 48, no. 6, pp. 1570–1579, Jun. 2002.

[4] D. MacKay and M. Postol, "Weakness of Margulis and Ramanujan-Margulis low-density parity check codes," *Electron. Notes Theor. Comp. Sci.*, vol. 74, 2003.

[5] T. Richardson, "Error-floors of LDPC codes," in *Proc. 2003 Allerton Conf. Communication, Control, and Computing*, pp. 1426–1435.

[6] L. Dolecek, Z. Zhang, V. Anantharam, M. Wainwright, and B. Nikolic, "Analysis of absorbing sets and fully absorbing sets of array-based LDPC codes," *IEEE Trans. Inf. Theory*, vol. 56, no. 1, pp. 181–201, Jan. 2010.

[7] P. O. Vontobel and R. Koetter, "Graph-cover decoding and finite-length analysis of message-passing iterative decoding of LDPC codes," CoRR, arxiv.org/abs/cs.IT/0512078.

[8] D. Divsalar and C. Jones, "Protograph based low error floor LDPC coded modulation," in *Proc. 2005 IEEE Military Commun. Conf.*, vol. 1, pp. 378–385.

[9] J. Lu and J. M. F. Moura, "Structured LDPC codes for high-density recording: large girth and low error floor," *IEEE Trans. Magnetics*, vol. 42, pp. 208–213, Feb. 2006.

[10] S. K. Chilappagari, S. Sankaranarayanan, and B. Vasic, "Error floors of LDPC codes on the binary symmetric channel," in *Proc. 2006 IEEE Int. Conf. Commun.*, pp. 1089–1094.

[11] S. Laendner and O. Milenkovic, "Algorithmic and combinatorial analysis of trapping sets in structured LDPC codes," in *Proc. 2005 Int. Conf. Wireless Networks, Commun., Mobile Comp.*, pp. 630–635.

[12] V. Savin, "Iterative LDPC decoding using neighborhood reliabilities," in *Proc. 2007 IEEE IEEE Int. Symp. Inform. Theory*, pp. 221–225.

[13] A. Casado, M. Griot, and R. Wesel, "Informed dynamic scheduling for belief-propagation decoding of LDPC codes," in *Proc. 2007 IEEE Int. Conf. Commun.*, pp. 932–937.

[14] E. Cavus and B. Daneshrad, "A performance improvement and error floor avoidance technique for belief propagation decoding of LDPC codes," in *Proc. 2005 IEEE Int. Symp. Pers., Indoor and Mobile Radio Comm.*, pp. 2386–2390.

[15] G. Kyung and C. Wang, "Exhaustive search for small fully absorbing sets and the corresponding low error-floor decoder," in *Proc. 2010 IEEE Int. Symp. Inform. Theory*, pp. 739–743.

[16] N. Varnica, M. P. C. Fossorier, and A. Kavcic, "Augmented belief propagation decoding of low-density parity-check codes," *IEEE Trans. Commun.*, vol. 55, no. 7, pp. 1308–1317, Jul. 2007.

[17] Y. Han and W. E. Ryan, "Low-floor decoders for LDPC codes," *IEEE Trans. Commun.*, vol. 57, no. 6, pp. 1663–1673, Jun. 2009.

[18] Z. Zhang, L. Dolecek, B. Nikolic, V. Anantharam, and M. Wainwright, "Lowering LDPC error floors by postprocessing," in *Proc. 2008 IEEE Global Telecom. Conf.*, pp. 1–6.

[19] J. Zhao, F. Zarkeshvari, and A. Banihashemi, "On implementation of min-sum algorithm and its modifications for decoding LDPC codes," *IEEE Trans. Commun.*, vol. 53, no. 4, pp. 549–554, Apr. 2005.

[20] T. Zhang, Z. Wang, and K. Parhi, "On finite precision implementation of LDPC codes decoder," in *Proc. 2001 IEEE ISCAS*, pp. 201–205.

[21] Z. Zhang, L. Dolecek, B. Nikolić, V. Anatharam, and M. Wainwright, "Design of LDPC decoders for improved low error rate performance: quantization and algorithm choices," *IEEE Trans. Wireless Commun.*, vol. 8, no. 11, pp. 3258–3268, Nov. 2009.

[22] Z. Zhang, "Design of LDPC decoders for improved low error rate performance," Ph.D. dissertation, Univ. of California at Berkeley, 2009.

[23] B. Butler and P. Siegel, "Error floor approximation for LDPC codes in the AWGN channel," in *Proc. 2011 Allerton Conf. Communication, Control, and Computing*, pp. 204–211.

[24] L. Dolecek, Z. Zhang, M. Wainwright, and V. Anatharam. "Evaluation of the low frame error rate performance of LDPC codes using importance sampling," in *Proc. 2007 IEEE Inform. Theory Workshop*, pp. 202–207.

[25] B. Frey, R. Koetter, and A. Vardy, "Signal-space characterization of iterative decoding," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 766–781, Feb. 2001.

[26] S. K. Planjery, D. Declercq, S. K. Chilappagari, and B. Vasic, "Multilevel decoders surpassing belief propagation on the binary symmetric channel," in *Proc. 2010 IEEE Int. Symp. Inform. Theory*, pp. 769–773.

[27] J. Chen, A. Dholakia, E. Eleftheriou, M. Fossorier, and X. Hu, "Reduced-complexity decoding of LDPC codes," *IEEE Trans. Commun.*, vol. 53, no. 8, pp. 1288–1299, Aug. 2005.

[28] IEEE Standard for Floating-Point Arithmetic, IEEE Standard 754-2008, Aug. 29, 2008.

[29] B. Butler and P. Siegel, "Numerical issues affecting LDPC error floors," in *Proc. 2012 IEEE Glob. Telecom. Conf.*, pp. 3201–3207.

[30] X. Hu, E. Eleftheriou, D. Arnold, and A. Dholakia, "Efficient implementations of the sum-product algorithm for decoding LDPC codes," in *Proc. 2001 IEEE Global Telecommun. Conf.*, vol. 2, pp. 1036–1036.

[31] G. Richter, G. Schmidt, M. Bossert, and E. Costa, "Optimization of a reduced-complexity decoding algorithm for LDPC codes by density evolution," in *Proc. 2005 IEEE Int. Conf. Commun.*, vol. 1, pp. 642–646.

[32] J. Chen and M. Fossorier, "Near optimum universal belief propagation based decoding of low-density parity check codes," *IEEE Trans. Commun.*, vol. 50, no. 3, pp. 406–414, Mar. 2002.

[33] W. E. Ryan and S. Lin, *Channel Codes: Classical and Modern*. Cambridge University Press, 2009.

[34] X. Zhang and P. H. Siegel, "Efficient algorithms to find all small error-prone substructures in LDPC codes," in *Proc. 2011 IEEE Global Telecom. Conf.*, pp. 1–6.

[35] D. J. C. MacKay, *Encyclopedia of Sparse Graph Codes*. Available: http://www.inference.phy.cam.ac.uk/mackay/codes/data.html

[36] J. Hamkins, "Performance of low-density parity-check coded modulation," *IPN Progress Report 42-184*, Feb. 2011. Available: http://ipnpr.jpl.nasa.gov/progress_report/42-184/184D.pdf

[37] X. Zhang and P. Siegel, "Quantized min-sum decoders with low error floor for LDPC codes," in *Proc. 2012 IEEE Int. Symp. Inform. Theory*, pp. 2871–2875.

[38] X. Zhang and P. Siegel, "Will the real error floor please stand up?" in *Proc. 2012 IEEE Int. Conf. Signal Process. Commun.*, pp. 1–5.

Xiaojie Zhang (S'05–M'13) received the B.S. degree in Electrical Engineering from Shanghai Jiao Tong University, Shanghai, China, in 2004, the M.S. degree in Electrical Engineering from Seoul National University, Seoul, Korea, in 2006, and the Ph.D. degree in Electrical Engineering from the University of California in San Diego, La Jolla, in 2012. From 2006 to 2008, he was a system engineer in Samsung Electronics, Suwon, Korea. Since 2013, he has been with Samsung Research America – Dallas conducting research and development on wireless cellular and local area networks. His research interests include wireless communication, multiuser scheduling, cross-layer communication system design, and error correction coding theory and its applications.

Paul H. Siegel (M'82–SM'90–F'97) received the S.B. and Ph.D. degrees in mathematics from the Massachusetts Institute of Technology (MIT), Cambridge, in 1975 and 1979, respectively.

He held a Chaim Weizmann Postdoctoral Fellowship at the Courant Institute, New York University. He was with the IBM Research Division in San Jose, CA, from 1980 to 1995. He joined the faculty at the University of California, San Diego in July 1995, where he is currently Professor of Electrical and Computer Engineering in the Jacobs School of Engineering. He is affiliated with the Center for Magnetic Recording Research where he holds an endowed chair and served as Director from 2000 to 2011. His primary research interests lie in the areas of information theory and communications, particularly coding and modulation techniques, with applications to digital data storage and transmission.

Prof. Siegel was a member of the Board of Governors of the IEEE Information Theory Society from 1991 to 1996 and from 2009 to 2011. He was re-elected for another three-year term in 2012. He served as Co-Guest Editor of the May 1991 Special Issue on "Coding for Storage Devices" of the IEEE TRANSACTIONS ON INFORMATION THEORY. He served the same TRANSACTIONS as Associate Editor for Coding Techniques from 1992 to 1995, and as Editor-in-Chief from July 2001 to July 2004. He was also Co-Guest Editor of the May/September 2001 two-part issue on "The Turbo Principle: From Theory to Practice" of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS.

Prof. Siegel was co-recipient, with R. Karabed, of the 1992 IEEE Information Theory Society Paper Award and shared the 1993 IEEE Communications Society Leonard G. Abraham Prize Paper Award with B.H. Marcus and J.K. Wolf. With J.B. Soriaga and H.D. Pfister, he received the 2007 Best Paper Award in Signal Processing and Coding for Data Storage from the Data Storage Technical Committee of the IEEE Communications Society. He holds several patents in the area of coding and detection, and was named a Master Inventor at IBM Research in 1994. He is an IEEE Fellow and a member of the National Academy of Engineering.