Iterative Decoding Beyond Belief Propagation

Shiva Kumar Planjery, Shashi Kiran Chilappagari, Bane Vasić
Department of ECE
University of Arizona
Tucson, AZ 85721, USA
Email: {shivap,shashic,vasic}@ece.arizona.edu

David Declercq, Ludovic Danjean
ETIS
ENSEA/UCP/CNRS UMR 8051
95014 Cergy-Pontoise, France
Email: {declercq,ludovic.danjean}@ensea.fr

***Abstract*—At the heart of modern coding theory lies the fact that low-density parity-check (LDPC) codes can be efficiently decoded by belief propagation (BP). The BP is an inference algorithm which operates on a graphical model of a code, and lends itself to low-complexity and high-speed implementations, making it the algorithm of choice in many applications. It has unprecedentedly good error rate performance, so good that when decoded by the BP, LDPC codes approach theoretical limits of channel capacity.**

However, this capacity approaching property holds only in the asymptotic limit of code length, while codes of practical lengths suffer abrupt performance degradation in the low noise regime known as the error floor phenomenon.

Our study of error floor has led to an interesting and surprising finding that it is possible to design iterative decoders which are much simpler yet better than belief propagation! These decoders do not propagate beliefs but a rather different kind of messages that reflect the local structure of the code graph. This has opened a plethora of exciting theoretical problems and applications. This paper introduces this new paradigm.

I. Introduction

Traditional message passing algorithms for decoding low-density parity-check (LDPC) codes are based on belief propagation (BP) [1], and operate on a graphical model of a code, known as the *Tanner graph* [2]. The BP, as an algorithm to compute marginals of functions on a graphical model, has its roots in the broad class of Bayesian inference problems [3]. While inference using BP is exact only on loop-free graphs (trees), it provides surprisingly close approximations to exact marginals on loopy graphs.

Exact inference in Bayesian belief networks is not only hard in general, but is hard even under strong restrictions of graphical model topology [4]. Thus, the research efforts are directed towards the design of efficient approximation algorithms [5]. It has been widely recognized that the algorithms that take into account the loopy structure of the graph can outperform algorithms that neglect the structure. However, with an exception of special-case or results of limited scope [6], [7], or empirical evidences [8], there have been no significant breakthroughs in this direction, especially in coding theory, and the decoder design has not moved further from the variations on the theme of generalized BP algorithms. One reason is that it is believed that such algorithms would have to closely approximate the exact inference, which would prohibitively increase the complexity.

A glimpse at the iterative decoders developed so far reveals a wide range of decoders of varying complexity. The simple binary message passing algorithms such as the Gallager A/B algorithms [9] occupy one end of the spectrum, while the BP lies at the other end of the spectrum. The gamut of decoders filling the intermediate space can simply be understood as the implementation of the BP (and variants) at different levels of precision. The thresholds for the quantization in these cases are optimized using density evolution [10]. Given the inapplicability of the asymptotic methods to finite length codes, it is hardly surprising that such quantized decoders exhibit poor performance leaving a large room for improvement.

It is apparent from the above discussion that the current decoders completely ignore the topology of the bit neighborhood as they still operate under the assumption that the graph is a tree. It is therefore appropriate to consider decoders in which the messages also convey information regarding the local neighborhood of a node in the graph.

In this paper, we shall introduce a novel approach for design of message-passing decoders that can outperform BP with a finite number of bits used to represent the messages. The bits in these messages do not represent beliefs but are chosen to reflect the local neighborhood of a particular node in the graph. In some cases, we show that only three bits are sufficient for the message-passing decoder to outperform BP in the error floor region and guarantee correction of a higher number of errors. We begin by providing a motivating example in Section II. We then describe a 2-bit decoder in Section III. In Section IV, we discuss the philosophy of our approach, and in Section V, we explain why these decoders have potential to outperform BP. We then provide a 3-bit decoder in Section VI along with numerical results in Section VII. We discuss relations to other decoders in Section VIII, and finally we provide conclusions and future work in Section IX.

II. A Motivating Example Using Two Bits

It was previously mentioned that BP is suboptimal for loopy graphs and decoders that utilize the local structure of the graph during message-passing can achieve better performance. A natural question that arises in this context is that what should the local message-passing rules be so that information about the neighborhoods can be inferred from the incoming messages and utilized in the determination of the outgoing messages? We begin with a simple example that involves

introducing errors on cycles for which the Gallager-A algorithm fails to decode, and demonstrating how a variable node involved in the cycle can capture its local neighborhood by the addition of just one extra bit in the messages.

Let us consider the decoding over binary symmetric channel (BSC) and assume that the all-zero codeoword is transmitted. This is valid since we consider only symmetric decoders as explained in [10]. Note that these assumptions shall be used during analysis of decoders throughout the paper. Consider the Tanner graph in which each variable node has degree three. Let V denote the set of all variable nodes and C denote the set of all check nodes in the Tanner graph. Assume that the channel introduced three errors and that the subgraph induced by the three variable nodes in error is a six cycle as shown in Fig. 1. Let $V_1 = \{v_1, v_2, v_3\}$ denote the set of variable nodes in the six cycle and let $C_1 = \{c_1, c_2, c_3\}$ and $C_2 = \{c_4, c_5, c_6\}$ denote the set of degree-two check nodes and degree-one check nodes in the subgraph respectively. Assume that no two check nodes in C_2 are connected to a common variable node in $V \backslash V_1$.

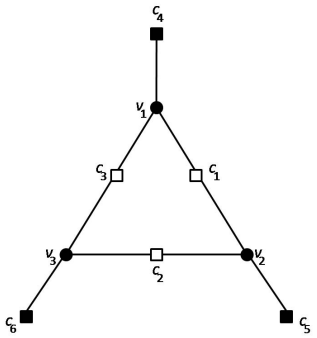

Fig. 1. Six-cycle present in the Tanner graph. The three variable nodes are initially in error.

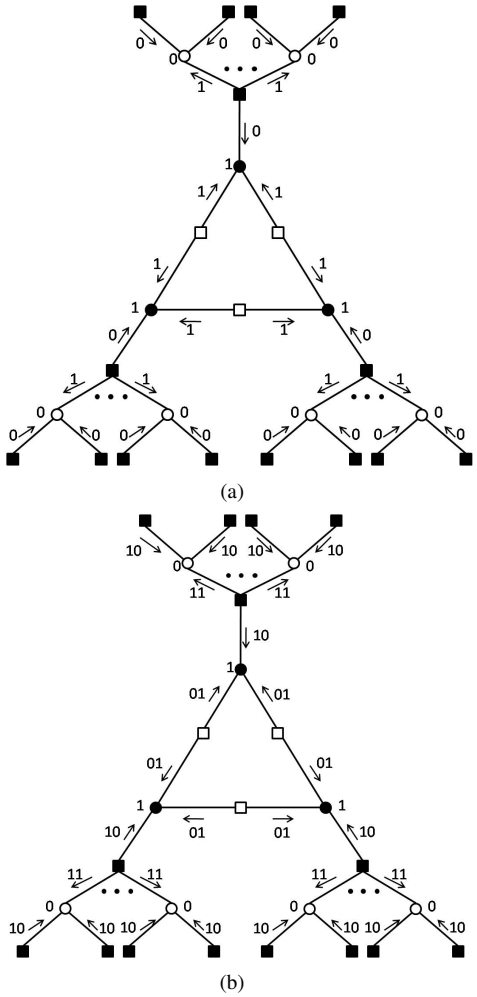

Fig. 2. Decoding on a six-cycle. Three variable nodes are initially incorrect, and all other variable nodes are correct. The variable node degree is three. Check node degree is arbitrary but only significant connections are shown. a) The messages as defined by Gallager A algorithm. The decoder is blind for the presence of a six-cycle, and cannot correct the above error configuration. b) Two-bit messages indicating the involvement of a variable node in a six-cycle.

A. Gallager-A decoder fails

Recall that the outgoing message at the variable node in the Gallager A algorithm is calculated as the majority of the incoming two messages and the received value. The outgoing message at the check node is simply the XOR of all the incoming extrinsic messages. Fig. 2(a) illustrates the message-passing process for the Gallager-A algorithm. Note that for figures throughout this paper, we shall use ●/○ to represent a variable node that is initially incorrect/correct, and a ■/□ to represent an odd/even degree check node. Following these rules, one can see that at the end of every iteration, all variable node neighbors of C_2 receive two message that coincide with their received value and one message that disagrees with the received value. Since, the neighborhood of C_2 consists of both correct and corrupt variable nodes, the algorithm cannot distinguish between the two based solely on the incoming messages. The variable nodes initially in error remain in error and all the other variable nodes remain uncorrupted. One realizes that this situation primarily arises due to the circulation of incorrect messages in the six cycle, but unfortunately the decoder cannot detect this scenario.

B. 2-bit decoder succeeds

Now consider adding one more bit to represent the messages in the decoder. Let the first bit (left-most bit) be used for conveying information on the local neighborhood and let the second bit (right-most bit) be used to denote the binary value of the variable node. In the initial iteration, the first bit of the outgoing message of a variable node is set to 0 and the second bit is set to the received value. At the check node, the first bit of the outgoing message is set to 1 if all the incoming messages have the first bit as 1; else it is set to 0. The second bit is simply the XOR of all the second bits in the incoming messages. In the second iteration at the variable node, the first bit of the outgoing message is set to 1 if the second bit of the two incoming messages agree with the received value; else, it is set to 0. The second bit is computed by taking the majority

of the second bits of the incoming messages along with the received value (as in Gallager-A). It can be seen that at the end of the second iteration, the variable nodes in V_1 receive two messages with the first bit being 0 and one message with the first bit being 1 (Fig. 2(b)).

Let $\mathcal{N}(U)$ denote the set of the neighbors of all nodes in a set U. Additionally, let $V_2 = \mathcal{N}(C_2)\backslash V_1$ and $C_3 = \mathcal{N}(V_2)\backslash C_2$ and assume no two checks in C_3 and C_1 share a common variable node in $V\backslash(V_1 \cup V_2)$. Based on the update rules defined so far, it can be shown that if the channel introduces only three errors, a variable node receives the above type of messages only if it is involved in a six cycle and the remaining two variable nodes in the six cycle are also in error. Moreover, the variable node can exactly identify which messages are coming from the six cycle and which are coming from the rest of the graph into the six cycle.

The decoder can now exploit this knowledge to intelligently choose the message passing rules so that the initially wrong variable node in the six cycle relies on the message coming from outside subgraph to start sending correct messages (messages containing the correct binary value in the second bit), whilst the remaining variable nodes outside the cycle continue sending correct messages.

III. 2-Bit Decoder For Column-Weight-Three Codes

We now completely specify the algorithm for the 2-bit decoder under which the error pattern in the previous example can be corrected. The message-passing rules for the variable node and check node can be considered as simple Boolean functions. These Boolean functions can also described as a set of conditions used to determing the outgoing messages. Note that for a degree-3 variable node, only two incoming messages are used for variable node update whereas all three messages are used in the decision rule.

2-Bit Decoding Algorithm

Initialization: The first bit of all outgoing messages from a variable node is set to 0, and the second bit is set to the received value.

Check node update: The first bit of the outgoing message is the AND of all the first bits in the incoming messages. The second bit is the XOR of all the second bits in the incoming messages.

Variable node update: Let $m_1 = \mathrm{a}_1\mathrm{a}_2$, $m_2 = \mathrm{b}_1\mathrm{b}_2$, denote the 2-bit incoming messages and $m_o = \mathrm{x}_1\mathrm{x}_2$ denote the 2-bit outgoing message at a variable node. $\mathrm{a}_1, \mathrm{a}_2$ denote the first and second bits in m_1. and similarly defined for $\mathrm{b}_1, \mathrm{b}_2$ and $\mathrm{x}_1, \mathrm{x}_2$. Let r denote the received value at the variabe node. Then do the following.

1) If $\mathrm{a}_1 = \mathrm{b}_1 = 1$, $\mathrm{a}_2 = \mathrm{b}_2$; set $\mathrm{x}_1 = 1$ and $\mathrm{x}_2 = \mathrm{a}_2$.
2) If $\mathrm{a}_1 = \mathrm{b}_1$, $\mathrm{a}_2 \neq \mathrm{b}_2$; set $\mathrm{x}_1 = 0$ and $\mathrm{x}_2 = r$.
3) If $\mathrm{a}_1 = \mathrm{b}_1 = 0$, $\mathrm{a}_2 = \mathrm{b}_2$; set x_2 to be majority of a_2, b_2 and r. If $\mathrm{a}_2 = \mathrm{b}_2 = r$, set $\mathrm{x}_1 = 1$, else set $\mathrm{x}_1 = 0$.
4) If $\mathrm{a}_1 \neq \mathrm{b}_1$, $\mathrm{a}_2 = \mathrm{b}_2$; set $\mathrm{x}_1 = 1$, $\mathrm{x}_2 = \mathrm{a}_2$.
5) If $\mathrm{a}_1 \neq \mathrm{b}_1$, $\mathrm{a}_2 \neq \mathrm{b}_2$; set $\mathrm{x}_1 = 0$. set $\mathrm{x}_2 = \mathrm{a}_2$ if $\mathrm{a}_1 = 1$ else set $\mathrm{x}_2 = \mathrm{b}_2$.

Decision rule: Let $\hat{u}$ denote the binary decision value at the variable node. Also in addition to m_1 and m_2, let $m_3 = \mathrm{d}_1\mathrm{d}_2$ denote the third message entering the variable node. $\hat{u}$ is the majority of a_2, b_2, d_2, and r. If there is no clear majority, then do the following.

1) If $\mathrm{a}_1 = \mathrm{b}_1 = \mathrm{d}_1$, then set $\hat{u}$ to be majority of a_2, b_2, d_2 (exclude r).
2) Else, let q be an auxillary Boolean variable and compute $q = \mathrm{a}_1 \oplus \mathrm{b}_1 \oplus \mathrm{d}_1$.
   If $q = 1$, set $\hat{u} = (\mathrm{a}_1 \cdot \mathrm{a}_2) \oplus (\mathrm{b}_1 \cdot \mathrm{b}_2) \oplus (\mathrm{d}_1 \cdot \mathrm{d}_2)$.
   If $q = 0$, set $\hat{u}$ to be majority of a_2, b_2, d_2.

$\oplus$ denotes XOR and $\cdot$ denotes AND.

It can be shown based on the update rules defined that this decoder is symmetric. Table I shows the Boolean map of the update rule at the variable node. Note the instance of the rule specified for incoming messages of '10' and '01' in Table I, which is critical for the variable node in the cycle to start sending correct messages.

m_1	m_2	r	m_o
00	00	0	10
00	00	1	00
00	10	0	10
00	10	1	10
00	01	0	00
00	01	1	01
00	11	0	01
00	11	1	01
10	10	0	10
10	10	1	10

m_1	m_2	r	m_o
10	01	0	00
10	01	1	00
10	11	0	00
10	11	1	01
01	01	0	01
01	01	1	11
01	11	0	11
01	11	1	11
11	11	0	11
11	11	1	11

TABLE I
BOOLEAN MAP USED AT THE VARIABLE NODE FOR THE 2-BIT DECODER

Although the example was specific to a three error pattern on the six cycle, the arguments can be generalized to cycles of arbitrary length and disjoint union of cycles. This leads to the following theorem pertaining to the defined 2-bit decoder for column-weight-three LDPC codes.

Theorem 1: Let V_1 denote the set of variable nodes involved in a cycle of length $2k$ contained in the Tanner graph, and let the subgraph induced by V_1 contain k degree-one checks. Let C_1 denote the set of degree-two checks and C_2 denote the set of degree-one checks in the subgraph respectively. Let $V_2 = \mathcal{N}(C_2)\backslash V_1$. If the channel introduces k errors exactly on the variable nodes involved in the cycle, and if no two checks in $\mathcal{N}(V_2)\backslash C_2$ and C_1 share a common variable node in $V\backslash(V_1 \cup V_2)$, then the 2-bit decoder successfully decodes this error pattern.

Sketch of proof: This follows by carrying out the message-passing process on the cycle similar to the previous example. Since all variable nodes outside the cycle are correct, using the previously defined update rules, it can be shown that the messages entering C_2 from nodes outside the cycle are '10' in the second iteration. Hence, at the end of the second iteration, every $v_i \in V_1$ will receive two messages of '01' from a check in C_1 and one message of '10' from a check in C_2. Because of the update rule, every $v_i \in V_1$ will send correct messages

to checks in C_1, and the decoder converges at the end of three iterations.

The above theorem validates the fact that the 2-bit decoder can perform better than Gallager-A which is not surprising. However, by adding more bits into the messages and cleverly choosing message-passing rules that incorporate the local neighbhorhood of a node, one would hope that the performance can be dramatically improved especially in the error floor region, so much so that it even outperforms floating-point algorithms such as min-sum and BP. In fact, we will later show that for certain classes of codes such as high-rate quasicyclic codes that are of great practical value, 3 bits are sufficient for the decoder to outperform BP provided the update rules are appropriately chosen.

IV. Philosophy of Our Approach

The previous example illustrated the use of one extra bit in the messages to capture the neighborhood of cycles, but then the next question that arises is how can additional bits be appended to the messages so that the variable nodes can utilize more complex neighborhoods (instead of just cycles) into the local computations without prohibitively increasing the complexity. At first sight, conveying the local structure of the Tanner graph by employing more bits appears complicated. However, as the previous example demonstrates, the topology surrounding the variable node can be compressed into a small number of bits in messages by using knowledge of what messages are possible for a given topology.

Also another desirable aspect that was illustrated in the previous example was that the variable nodes in the six cycle did not know *a priori* that they were part of the cycle and this was only inferred from their incoming messages during decoding. This is important since we would like decoders that are not specifically designed for any particular code, but rather perform well on any general code.

A. Adding bits to design increasingly successful decoders

The central idea behind deriving good message-passing rules that account for the local neighborhood can be described as follows. We first identify a set of potentially troublesome topological structures that are known for traditional message-passing decoders such as Gallager-A/B or BP. These troublesome structures generically termed as trapping sets [11], are the primary reason for the error floor phenomenon in LDPC codes and can be present in any finite-length code. A standard notation for a trapping set is (a, b) where a is the set of variable nodes that eventually failed to decode [11] and b is the number of odd-degree check nodes in the sub-graph induced by a. Parameters (a, b) do not completely describe a trapping set but we will use this notation because of traditional reasons. Given a list of trapping sets, the main goal would be to derive rules that can correct a majority of these trapping sets with minimal number of bits used in the messages, by analyzing the decoding on their induced subgraphs. Through the analysis of decoding on these subgraphs, more and more extra bits can be can be progressively added to the messages such that the nodes in the subgraph capture a more complicated local neighborhood in these trapping sets and the rules can be derived to eventually correct all of them. For instance, the 2-bit decoder is guaranteed to correct all (a, a) trapping sets of Gallager-A because of Theorem 1.

In order to provide a clearer understanding of our approach, we shall illustrate with an example how a third bit can be added to the messages of a 2-bit decoder so that a local neighborhood that is larger and more complex than a cycle can be captured by a variable node. The example we shall use will be in the same spirit of the previous example but now we consider a more complicated topological structure for which the 2-bit decoder fails to decode. We will later elaborate on the concept of decoding beyond BP which is the ultimate aim.

B. Using three bits to decode a trapping set

Consider the subgraph induced by a $(5, 3)$ trapping set of the 2-bit decoder as shown in Fig. 3, and let this subgraph be contained in the Tanner graph of a column-weight-three code. Assume that the channel introduces five errors exactly on the variable nodes in this subgraph and the remaining variable nodes that are outside the subgraph receive correct values (which is 0). Let V denote the set of all variable nodes and C denote the set of all check nodes in the Tanner graph. Let $V_1 = \{v_1, v_2, v_3, v_4, v_5\}$ denote the set of variable nodes in the subgraph and let $C_1 = \{c_1, c_2, c_3, c_4, c_5, c_6\}$ and $C_2 = \{c_7, c_8, c_9\}$ denote the sets of degree-two check nodes and degree-one check nodes in the subgraph respectively. While performing decoding on this subgraph, we shall assume that the neighborhood of this subgraph in the Tanner graph is such that the messages being propagated within the subgraph do not in any way influence the messages that are entering the subgraph from outside the subgraph. A more rigrous explanation to such a condition will be explained in Section V-A. Under these assumptions, we will in fact show that this structure is indeed a trapping set for the 2-bit decoder.

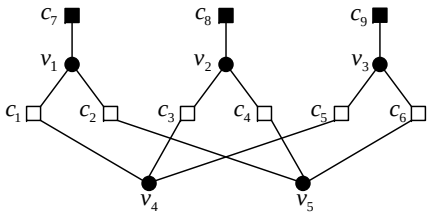

Fig. 3. Subgraph induced by $(5, 3)$ trapping set that is present in the Tanner graph. The five variable nodes are initially in error.

1) 2-bit decoder fails: Consider the decoding on the subgraph with the 2-bit decoding algorithm defined in the previous section. Fig. 4 illustrates the message-passing process on the subgraph for the first two iterations. In the beginning of iteration 1, all variable nodes in V_1 will send '01' to their neighboring checks as they are initially wrong, and all remaining variable nodes in $V \backslash V_1$ will send '00' as they are initially correct. At the end of iteration 1, nodes v_1, v_2, v_3

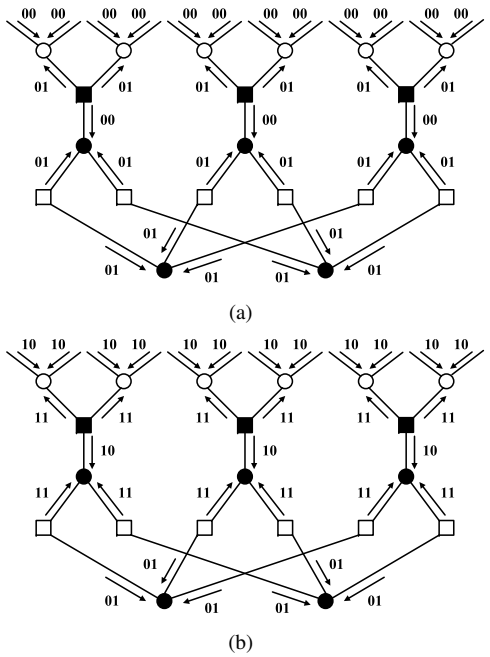

Fig. 4. Decoding on subgraph induced by a $(5, 3)$ trapping set using 2-bit decoder which fails to converge. a) The messages passed at the end of iteration 1. b) Messages passed at the end of iteration 2

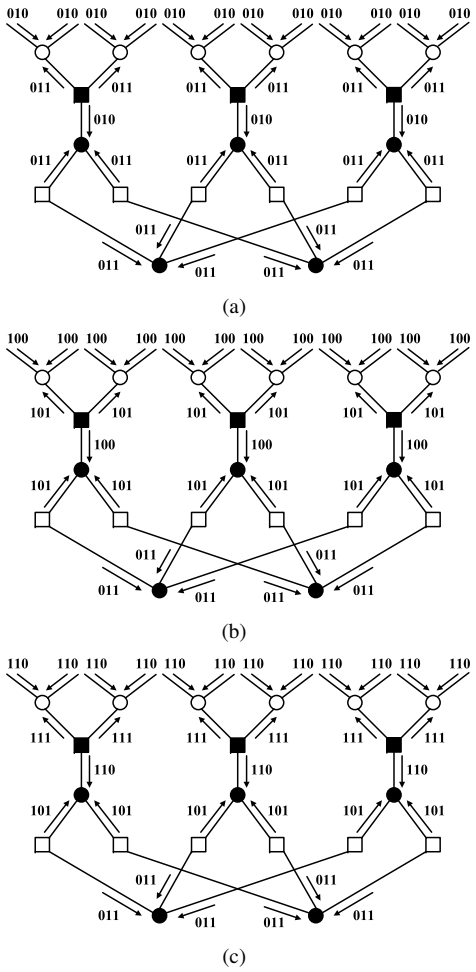

Fig. 5. Decoding on subgraph induced by a $(5, 3)$ trapping set using 3-bit decoder. a) Messages passed at the end of iteration 1. b) Messages passed at the end of iteration 2 . c) Messages passed at the end of iteration 3.

will all receive two messages of '01' and one message of '00', whereas nodes v_4 and v_5 will receive three messages of '01', and nodes in V_2 will receive two messages of '00' and one message of '01', as shown in Fig. 4(a). In the beginning of second iteration, nodes v_1, v_2, v_3 will send messages of '01' to checks in C_1, and nodes v_4 and v_5 will all send '11' to their neighboring checks. At the end of iteration 2, nodes v_1, v_2, v_3 receive two messages of '11' and one message of '10', whereas nodes v_4 and v_5 will again receive three messages of '01', as shown in Fig. 4(b) . Since the update rule at the variable for the instance of '10', '11', and $r = 1$, is '01', the same type of messages are passed again as in iteration 2 and the decoder fails to converge. Notice that the 2-bit decoder failed to converge since the two bits in the messages were not enough to capture the local neighborhood of this particular topology, i.e, it was equipped to only capture local neighborhoods of cycles.

Remark: The Gallager-A algorithm also fails on the $(5, 3)$ trapping set with just three variable nodes v_1, v_2, and v_3 initially in error. Whereas the 2-bit decoder fails only if all nodes in V_1 are initially in error. So the 2-bit decoder is still better than Gallager-A .

2) 3-bit decoder succeeds: Now consider adding an extra bit in the messages and define the message-passing rules as follows. In iteration 1, for every outgoing message of a variable node, let the first bit be set to 0, second bit be set to 1 and the third bit be set to the received value. Then we see that all nodes in V_1 send '011' and all nodes in $V \backslash V_1$ send '010' to their neighboring checks. At the check node, set the first bit of the outgoing message to 1 only if all incoming messages are 1; else set to 0. Set the second bit of the outgoing message to be the second bit of the incoming message with '0' as its first bit. Set the third bit to be the XOR of third bits of all the incoming messages. Then at the end of iteration 1, nodes v_1, v_2, v_3 will receive two messages of '011' and one message of '010', and nodes v_4 and v_5 will receive three messages of '011', as shown in Fig. 5(a).

If we continue to define rules for more iterations and the message-passing process is continued up to iteration 3, since there are many different 3-bit messages that can possibly be passed (eight as opposed to only four for the 2-bit decoder) based on the rules defined, the nodes v_1, v_2, and v_3 (under certain assumptions that will be explain in the next section), can infer that they are part of this subgraph from the type of incoming messages they receive in each iteration. In essence, these variable nodes are able to infer the topology with the help of an extra third bit in the messages, and now the message-passing rules can be derived so that nodes v_1, v_2, and v_3 start sending correct messages. Note however that each of the nodes v_4 and v_5 will still be blind to its local neighborhood as all its neighbors are initially wrong. Therefore, a key point to note here is that the message-passing rules must be derived so

that the nodes v_1, v_2, and v_3 infer their local neighborhoods and start propagating correct messages in the fewest possible iterations to inhibit the propagation of wrong messages by nodes v_4 and v_5.

We note that there are possibly many different 3-bit decoders that can correct the above $(5, 3)$ trapping set. However, not all of them are good decoders especially when considering decoders beyond BP. The decoder must be designed to correct many potentially troublesome subgraphs and the $(5, 3)$ trapping set is only one such structure. So one aspect of our approach is identifying the relevant trapping sets and using trapping set ontology [16] that is known for existing message-passing decoders to design good update rules. The other important aspect is related to the speed of convergence of such decoders which will be addressed in Section V-B.

So far we have discussed how bits can be added to the messages to incorporate the local structure in the message-passing rule and therefore improve the ability of the decoder to correct trapping sets. We will now attempt to explain how these decoders have potential to even outperform BP with the help of a concept called isolation assumption.

V. Decoding Beyond BP and Isolation Assumption

From our discussion of the two examples, it was established that the underlying strategy behind deriving good message-passing rules lied in the analysis of subgraphs induced by trapping sets. However, note that in both the examples, the decoding was carried out under certain assumptions, and it is quite likely that those assumptions will not be valid on a Tanner graph of a practical code. A pivotal notion that enables us to analyze decoders on subgraphs induced by trapping sets, is the concept of *isolation assumption* which was also inherently used in the previous two examples. We shall now discuss this concept in greater detail.

A. Isolation assumption

While considering decoding on a particular trapping set, in order to verify whether a given decoder succeeds on the trapping set, it is in fact necessary not only to know the subgraph induced by the trapping set, but also the neighborhood of the induced subgraph and messages coming from this neighborhood. Since this neighborhood can be different for different Tanner graphs, the derivation of the message-passing rules can become very complex. Therefore, to facilitate the design process, we work under the assumption that the messages to the nodes of the trapping set from the nodes outside the induced subgraph of the trapping set are known. We coined this the *isolation assumption* to signify the fact that the trapping set can be considered in isolation from the rest of the graph and can be analyzed as an independent entity.

A more formal definition taken from our work in [13] is provided in the Appendix that uses the notion of computation trees [14], [15]. It is important to note that this is different from the independence assumption considered by Gallager [9] and an explanation for this can also be found in [13]. This concept gives rise to the *isolation theorem* that helps us determine the messages coming from the neighborhood of the induced subgraph. The theorem is also included in the Appendix.

Note that the isolation assumption is independent of the decoder and can be considered as a property of the Tanner graph in relation to a particular subgraph. This assumption validates up to how many iterations a subgraph can be considered isolated from its Tanner graph for decoding. This is precisely the reason why the speed of convergence of the decoder becomes important while decoding on an isolated subgraph, since for Tanner graphs of practical codes, the isolation assumption becomes hard to satisfy for large number of iterations by the subgraph. If the isolation assumption is violated by the subgraph for a particular Tanner graph, then the decoder is not guaranteed to correct the error pattern in the subgraph (even though a rule was derived to correct it).

B. BP and min-sum fail due to slow convergence

The concept of isolation assumption can now be used to explain why these newly defined message-passing decoders can correct certain error patterns that are uncorrectable even by floating-point algorithms such as BP and min-sum. An intuitive reason for this improvement can be provided as follows.

Consider the subgraph induced by an uncorrectable low-weight error pattern of BP or min-sum along with sufficient neighborhood of this subgraph that is present in the Tanner graph. Perform decoding on this subgraph under the isolation assumption with the help of computation trees and the isolation theorem. Since the nodes in the graph for BP or min-sum decoders do not have any knowledge of their local neighborhood, the decoder can take several iterations to converge on the isolated subgraph. Now in the actual Tanner graph that contains this subgraph, the isolation assumption of the subgraph will be violated within few iterations and therefore, this is the reason why the BP or min-sum decoders fail to decode on such error patterns. This was found to be especially true for codes with dense graphs. On the other hand, in the case of the 3-bit decoder, the variable nodes can propel the decoder through their update rules to converge much faster on the same error pattern, so that the decoder converges before the isolation assumption is violated on the subgraph. Hence, our message-passing decoders such as the 3-bit decoder can successfully correct such error patterns and thereby guarantee a higher error-correction capability than BP or min-sum. Our numerical results which we will show in Section VII, also support this argument, since we considered codes that have relatively dense graphs such as high-rate quasicyclic codes, and on such codes a simple 3-bit decoder was enough to outperform BP and min-sum.

VI. 3-Bit Decoder For Column-Weight-Three Codes

We now provide a particularly good 3-bit decoder that was derived by considering a systemtatic hierarchy of trapping sets called trapping set ontology [16]. The update rules at the variable nodes and check nodes are judiciously designed

Boolean functions that can guarantee correction of higher number of errors than BP or min-sum over the BSC. Some important criteria that were considered in the derivation of the rules are increase in critical number [17] and convergence in fewest possible iterations.

The first two bits in the messages are used to capture the local neighbhorhood and the third bit (right-most bit) is used to denote the binary value of the variable node. Also for this particular decoder, the messages '001' and '000' are treated as the same, since they can be regarded as erasures as in Gallager-E algorithm [10]. For convenience we just use one message '000'.

3-bit Decoding Algorithm

Initialization: For all outgoing messages passed from a variable node, the first bit is set to 0, the second bit is set to 1, and the third bit is set to the received value.

Check node update: Let $\mathrm{y}_1, \mathrm{y}_2, \mathrm{y}_3$ denote the first, second and third bits of the outgoing message of a check node respectively. The set of update rules that define the Boolean function for the check node are as follows.

1) y_3 is the XOR of the third bits of all the incoming messages.
2) y_1 is the AND of the first bits of all the incoming messages.
3) a) If all the incoming messages have their first bit as 1, then y_2 is the AND of the second bits of all the incoming messages.
   b) Else consider only the incoming messages that have their first bit as 0. Then y_2 is the AND of all the second bits corresponding to only these messages.
   c) If there is only one incoming message having a first bit as 0, then y_2 is the same value as the second bit of that particular message.

Variable node update: Let $m_1 = \mathrm{a}_1\mathrm{a}_2\mathrm{a}_3$, $m_2 = \mathrm{b}_1\mathrm{b}_2\mathrm{b}_3$, and $m_o = \mathrm{x}_1\mathrm{x}_2\mathrm{x}_3$ denote the incoming messages and outgoing message of a variable node respectively. $\mathrm{a}_1, \mathrm{a}_2, \mathrm{a}_3$ denote the first, second, and third bits of m_1, and similarly defined for bits of m_2 and m_3. Let r denote the received value at the variable node. We can similarly define the set of rules for the variable node update as a set of conditions. For example, the following conditions specify the rule when the incoming messages are either '110' or '111', and the received value is 0 or 1.

1) If $\mathrm{a}_1 = \mathrm{b}_1 = 1$ and $\mathrm{a}_2 = \mathrm{b}_2 = 1$, set $\mathrm{x}_2 = 1$, $\mathrm{x}_3 = r$.
2) In addition, if $\mathrm{a}_3 = \mathrm{b}_3$, set $\mathrm{x}_1 = 1$; Else, set $\mathrm{x}_1 = 0$.

Since there are many cases that need to be considered, we shall not provide the entire set of conditions required to determine the bits of the outgoing message. Instead, we provide the complete Boolean map used at the variable node in the form of Table II. It is quite possible that with deeper introspection into the boolean map, simple boolean expressions can be derived to determine the bits of the outgoing message.

Reverting back to the the example involving the $(5, 3)$ trapping set, note that the instance of the update rule defined for incoming messages of '100', '101' and $r = 1$ which gives '011' in the second iteration , and the instance of the update rule for incoming messages of '110','101', and $r = 1$ which still gives '011' in the third iteration, as shown in Fig. 5(b) and Fig. 5(c) , are crucial for the nodes v_1, v_2, and v_3 to inhibit the propagation of wrong messages by v_4 and v_5 and start sending correct messages to them after third iteration.

m_1	m_2	r	m_o
010	010	0	100
010	010	1	000
010	100	0	100
010	100	1	010
010	110	0	110
010	110	1	100
010	000	0	010
010	000	1	000
010	011	0	010
010	011	1	011
010	101	0	011
010	101	1	101
010	111	0	101
010	111	1	111
100	100	0	110
100	100	1	100
100	110	0	110
100	110	1	110
100	000	0	100
100	000	1	010
100	011	0	100
100	011	1	010
100	101	0	010
100	101	1	011
100	111	0	101
100	111	1	101
110	110	0	110
110	110	1	110
110	000	0	110
110	000	1	100
110	011	0	110
110	011	1	100
110	101	0	100
110	101	1	100
110	111	0	010
110	111	1	011
000	000	0	010
000	000	1	011
000	011	0	000
000	011	1	011
000	101	0	011
000	101	1	101
000	111	0	101
000	111	1	111
011	011	0	000
011	011	1	101
011	101	0	101
011	101	1	101
011	111	0	101
011	111	1	111
101	101	0	101
101	101	1	111
101	111	0	111
101	111	1	111
111	111	0	111
111	111	1	111

TABLE II
BOOLEAN MAP USED AT THE VARIABLE NODE FOR THE 3-BIT DECODER.

Alternatively, Boolean functions can be described algebraically where the messages take values (or levels) from a finite set [13]. In this case, the actual value of the message contains information about the local structure of the node rather than likelihood of particular bits. There is a one-to-one correspondence between the algebraic form and boolean function form. The algebraic description could sometimes be more convenient while deriving good update rules especially when considering higher number of bits or levels allowed in the messages. Also the decision rule which was not specified in the algorithm can easily be derived using the algebraic description. For more details on the algebraic description, refer to our work in [13].

We now provide numerical results for this 3-bit decoder and compare it with BP and min-sum.

VII. NUMERICAL RESULTS

Simulations were carried out on four different column-weight-three codes: 1) $n = 155$, $R = 0.4$, Tanner code, 2) $n = 768$, $R = 0.75$, quasicyclic code with $d_{min} = 12$, 3) a $n = 4085$, $R = 0.82$, MacKay code, and 4) $n = 1503$, $R = 0.668$, array code with $d_{min} = 16$. Frame error rates

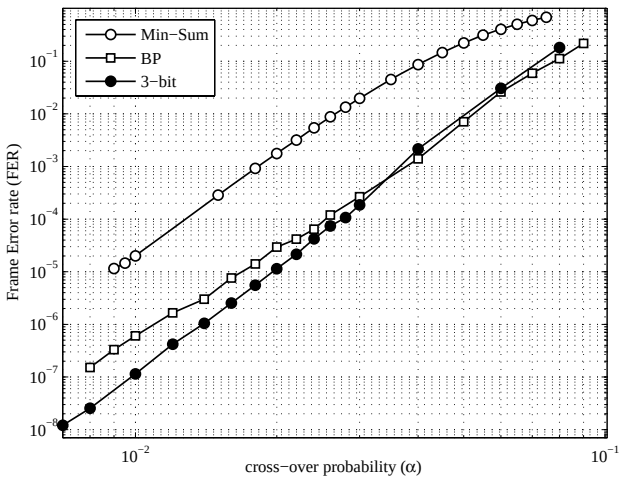

Fig. 6. FER results on the $n = 155$, $R = 0.4$, Tanner code

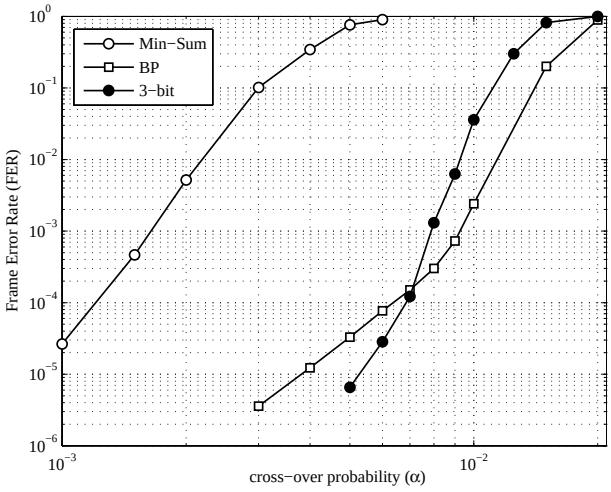

Fig. 8. FER results on the $n = 4085$, $R = 0.82$, MacKay code

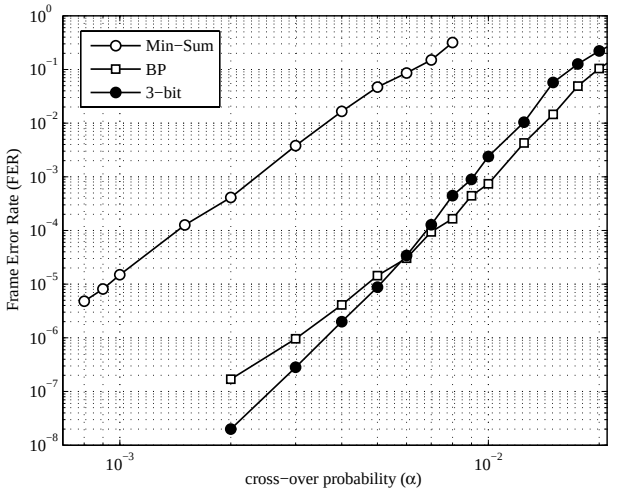

Fig. 7. FER results on the $n = 768$, $R = 0.75$, Quasicyclic code

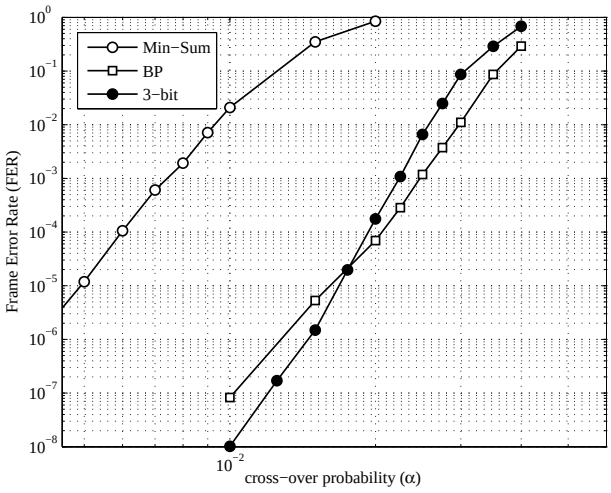

Fig. 9. FER results on the $n = 1503$, $R = 0.668$, array code

(FER) are plotted as a function of the cross-over probability α of the BSC. n denotes the length of the code, R denotes the code rate, and d_{min} denotes the minimum distance of the code. The codes were chosen to cover a broad spectrum of LDPC codes in order to validate our approach. The Tanner code is well-understood and has been analyzed for many different decoders. The high-rate quasicyclic code was chosen since the error floor problem is much more challenging for high-rate codes. A MacKay code was chosen as an example of a high-rate random code. A moderate length high-rate array code with fairly reasonable minimum distance was chosen since these codes have high practical value as they enable simple implementations. Fig. 6, Fig. 7, Fig. 8, Fig. 9 show the FER results. The parity check matrices of all four of these codes can be found in [19].

The numerical results epitomize the underlying philosophy of our approach as we see that the 3-bit decoder outperforms BP and min-sum in the error floor region for all the four codes. Notice the difference in slopes in the FER curves which is related to number of fixed errors a decoder can correct [18]. By using just three bits in the messages and deriving message-passing rules that take the local neighborhood into consideration, we see that these decoders are able to achieve this performance improvement with only a fraction of the complexity of the BP and min-sum decoders.

Although we did not provide specific examples, it can be shown that this 3-bit decoder is capable of correcting certain low-weight error patterns that are uncorrectable by BP and min-sum depending on the structure of the code. In fact for the Tanner code, the 3-bit decoder guarantees correction of

all error patterns upto 5 errors. An interesting remark to note is that the highly complex linear programming (LP) decoding fails to correct all 5-errors on the Tanner code, whereas the simple 3-bit decoder is able to. Also, the 3-bit decoder did not fail for any 4-error patterns on the $n = 768$, $R = 0.75$, quasicyclic code whereas the BP and min-sum decoders failed on such error patterns in the region of simulation in Fig. 7. For a specific example on how the 3-bit decoder is able to correct such error patterns, refer to [13]. In the next section, we shall discuss the relations of our message-passing decoders to other relevant decoders and highlight key differences in our approach from existing approaches.

VIII. Relations to other decoders

A. Quantized message-passing decoders

It can be recognized that the message-passing decoders considered in this paper are related to the class of quantized decoders. There have been many significant works related to the design and analysis of quantized decoders as well as low-complexity implementations of BP-based decoders. Some of the notable works include (but not limited to) the quantized decoders (such as Gallager-E) proposed by Richardson and Urbanke [10], low-complexity BP decoders by Chen *et al.* [20], and by Fossorier, Mihaljevic and Imai [21], quantized BP decoders by Lee and Thorpe [22], and quantized min-sum decoders by Smith, Kschischang and Yu [23]. A key distinction that is to be noted between our approach and all these aforementioned works is that their primary objective is to approach BP rather than outperform BP, since they are all based on asymptotic analytical methods. In addition, they do not guarantee good performance on a finite-length code in the low noise region.

B. Modifying decoders to reduce error floor

Efforts to reduce error floors by modifying the decoder have also received significant attention and have doubtlessly provided valuable insights into the working of iterative decoding. Augmented belief propagation in which selected initial channel messages are saturated after sufficient number of iterations has been suggested as an effective way to reduce error floor [24]. Adapting the parity-check matrix at each iteration or running the decoder in parallel on equivalent parity-check matrices and combination of both has also proved to be useful [25]. Adding redundant rows to the parity-check matrix has also been considered in [26]. Multi-stage decoding, in which decoders of increasing complexity are used in a sequential manner so as to result in progressive improvement in the FER performance was proposed by Wang, Yedidia, and Draper in [27].

Many of these techniques have certainly proved to be useful for improving the performance of a particular decoder or a particular code. However, most of the approaches are either complex or restricted to a specific class of codes, and some methods consist of searching Tanner graphs for subgraphs that are believed to be harmful, where the harmfulness of such subgraphs are not proven or are only restricted to certain cases. Our approach differs from the above works in that no a priori assumptions are made about harmfulness of subgraphs.

C. Statistical mechanics methods

Methods to systematically account for the presence of loops in BP was studied in the most general setting by Yedidia, Freeman and Weiss [28] and later by Chertkov and Chernyak [29] and by Lu, Measson and Montanari [30].

Yedidia *et al.* [28] showed how the free energy hierarchy approximations can be improved using generalized belief propagation. Chertkov *et al.* [29] used loop calculus to improve the approximations provided by the BP and also demonstrated how it can be used to reduce error floors. More recently, Lu *et al.* [30] proposed a tree pruning (TP) decoding algorithm, which prunes the computation tree so as to provide a closer approximation to the true marginal compared to BP. In this way, it systematically accounts for loops and provides a way to successively improve the approximations from BP to the optimal decoder.

These methods while supported by strong theoretical justifications are still far from finding their way into practical decoders for moderate length codes.

IX. Conclusions and Future Work

In this paper, we presented a new methodology for designing low-complexity message-passsing decoders that utilized bits in the messages wherein the rules were derived using trapping set ontology of traditional decoders. From the results, we see that there is a certain aspect of universality in the 3-bit decoder; the 3-bit decoder is good on a variety codes, which is highly desirable. Although the simulation results were performed on only certain classes of codes such as high-rate quasicyclic codes, we surmise that by considering more bits in the messages, better decoders can be designed that have potential to outperform BP even on codes with moderate rates.

Our future work includes further investigation into the properties of these decoders, deriving bounds on error-correction capability, providing conditions on the graphs based on the isolation assumption that guarantees higher correction of errors than BP, and possibly relating all these to bounds on the number of bits required to guarantee correction of fixed number of errors in a fixed number of iterations. We would also like to investigate the concept of time-varying decoders where the message-passing rules are allowed to change from iteration to iteration. A small comment be noted regarding the FER results of the Tanner code is that in addition to the message-passing rules defined for the 3-bit decoder in the Section VI, a slightly modified rule that was also derived using trapping sets, was used sequentially (whenever the previous failed) to correct certain 5-error patterns of the Tanner code. We do not provide the details of this modified rule but it shall be explained in our future work when we obtain more results on time-varying decoders.

Appendix

Let $G = \{V \cup C, E\}$ be the Tanner graph. Let H be the induced subgraph of a trapping set (a, b) contained in G with

variable node set $P \subseteq V$ and check node set $W \subseteq C$. Let $\mathcal{N}(u)$ denote the set of neighbors of a node u. Let $T_i^k(G)$ be the computation tree of graph G corresponding to a messagee-passing decoder enumerated for k iterations with variable node $v_i \in V$ as its root. Let $P' \subseteq P$ denote the set of variable nodes in subgraph H that have degree-one check nodes as its neighbors. Let $W' \subseteq W$ denote the set of degree-one check nodes in the subgraph H. Let supp($\mathbf{r}$) denotes the set of all variable nodes that received wrong values (which is 1) from the BSC. Let Φ_v denote the Boolean map used at the variable node, which is a function of the incoming messages and the received value.

Definition 1: A vertex $w \in T_i(G)$ is said to be a *descendant* of a vertex $u \in T_i(G)$ if there exists a path starting from vertex w to the root v_i that traverses through vertex u. The set of all descendants of the vertex u in $T_i(G)$ is denoted as $\mathcal{D}(u)$. For a given vertex set U, $\mathcal{D}(U)$ (with some abuse of notation) denotes the set of descendants of all $u \in U$.

Definition 2 (Isolation assumption): Consider the computation tree $T_i^k(G)$ with the root $v_i \in P'$. Pick a $c_j \in N(v_i) \cap W'$ which is a degree-one check node in H. If the condition $\mathcal{D}(c_j) \cap \mathcal{D}(\mathcal{N}(v_i) \setminus c_j) = \emptyset$ is satisfied $\forall c_j \in N(v_i) \cap W'$, and if for any two check nodes $c_r, c_s \in W \setminus W'$, $\mathcal{D}(c_r) \cap \mathcal{D}(c_s) \subseteq (P \cup W)$, then the tree $T_i^k(G)$ is said to be isolated. If $T_i^k(G)$ is isolated $\forall v_i \in P'$, then the subgraph H is said to satisfy the isolation assumption for k iterations.

Theorem 2 (Isolation theorem): If $\mathbf{r}$ is input to the message-passing decoder from the BSC such that supp($\mathbf{r}$) $\in P$, and if H satisfies the isolation assumption for k iterations, then for each $c_j \in W'$, the message from c_j to its neighbor in H in the l^{th} iteration denoted by μ_l, is determined as the output of $\Phi_v(\mu_{l-1}, \mu_{l-1}, 0) \ \ \forall l \leq k$.

For details on the proof, refer to [13]

ACKNOWLEDGMENT

This work is funded by the NSF under Grants CCF-0634969, IHCS-0725405 and CCF-0830245. Authors would like to thank Dzung Viet Nguyen for providing the high-rate array code used in our simulation.

REFERENCES

[1] J. Pearl, *Probablisitic Reasoning in Intelligent Systems*. San Francisco, CA: Kaufmann, 1988.

[2] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inform. Theory*, vol. 27, no. 5, pp. 533–547, May 1981.

[3] B. J. Frey, *Graphical models for machine learning and digital communication*. Cambridge, MA, USA: MIT Press, 1998.

[4] G. F. Cooper, "The computational complexity of probabilistic inference using Bayesian belief networks (research note)," *Artif. Intell.*, vol. 42, no. 2-3, pp. 393–405, 1990.

[5] P. Dagum and M. Luby, "Approximating probabilistic inference in Bayesian belief networks is NP hard," *Artif. Intell.*, vol. 60, no. 1, pp. 141–153, 1993.

[6] Y. Weiss and W. T. Freeman, "Correctness of belief propagation in Gaussian graphical models of arbitrary topology," *Neural Computation*, vol. 13, no. 10, pp. 2173–2200, 2001.

[7] D. Weitz, "Counting independent sets up to the tree threshold," in *STOC 06: Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*. New York, NY, USA: ACM, pp. 140–149, 2006.

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

[9] R. G. Gallager, *Low Density Parity Check Codes*. Cambridge, MA: M.I.T. Press, 1963.

[10] T. Richardson and R. Urbanke, "Capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inform. Theory* , vol 47, pp. 599–618, Feb. 2001.

[11] T. Richardson, "Error floors of LDPC codes," in *Proc. 41st Annual Allerton Conference on Communications, Control and Computing*, 2003.

[12] S. K. Chilappagari, S. Sankaranarayanan, and B. Vasic, "Error floors of LDPC codes on the binary symmetric channel," in *Proc. IEEE International Conference on Communications (ICC 06)*, vol. 3, Istanbul, Turkey, pp. 1089–1094, 2006.

[13] S. K. Planjery, D. Declercq, S. K. Chilappagari, and B. Vasic, "Multilevel decoders surpassing belief propagation on the binary symmetric channel," Preprint [Online]. Available: http://arxiv.org/abs/1001.3421

[14] N. Wiberg, "Codes and decoding on general graphs," Ph.D. dissertation, Univ. Linkoping, Sweden, Dept. Elec. Eng., 1996.

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

[16] B. Vasic, S. K. Chilappagari, D. V. Nguyen, and S. K. Planjery, "Trapping set ontology," *Proc. 47th Annual Allerton Conference on Communications, Control, and Computing*, Sept. 2009. [Online]. Available: http://www.ece.arizona.edu/~vasiclab/Projects/CodingTheory/TrappingSetOntology.pdf

[17] S. K. Chilappagari, A. R. Krishnan, and B. Vasic, "LDPC codes which can correct three errors under iterative decoding," in *Proc. IEEE Information Theory Workshop*, pp. 406–410, May 2008.

[18] M. Ivkovic, S. K. Chilappagari, and B. Vasic, "Eliminating trapping sets in low-density parity-check codes by using Tanner graph covers," *IEEE Trans. Inform. Theory*, vol. 54, no. 8, pp. 3763–3768, 2008.

[19] "Error Floors of LDPC Codes." [Online]. Available: http://www.ece.arizona.edu/~vasiclab/Projects/CodingTheory/ErrorFloorHome.html

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

[21] M. Fossorier, M. Mihaljevic, and H. Imai, "Reduced complexity iterative decoding of low-density parity check codes based on belief propagation," *IEEE Trans. Commun.*, vol. 47, no. 5, pp. 673–680, May 1999.

[22] J. K. Lee and J. Thorpe, "Memory-efficient decoding of LDPC codes," in *Proc. International Symposium on Information Theory (ISIT 2005)*, Adelaide, Australia, pp. 459–463., Sept. 2005.

[23] B. Smith, F. R. Kschischang and W. Yu, "Low-density parity-check codes for discretized min-sum decoding," in *Proc. 23rd Biennial Symposium on Communications*, pp. 14–17, 2006.

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

[25] T. Hehn, J. B. Huber, S. Laendner, and O. Milenkovic, "Multiple-bases belief-propagation for decoding of short block codes," in *Proc. IEEE International Symposium on Information Theory (ISIT 07)*, Nice, France, pp. 311–315, June 2007.

[26] S. Laendner, T. Hehn, O. Milenkovic, and J. Huber, "The trapping redundancy of linear block codes," *IEEE Trans. Inform. Theory*, vol. 55, no. 1, pp. 53–63, Jan. 2009.

[27] Y. Wang, J. S. Yedidia, and S. C. Draper, "Multi-stage decoding of LDPC codes," in *Proc. IEEE International Symposium on Information Theory (ISIT 09)*, Seoul, Korea, pp. 2151–2155, July 2009.

[28] J. S. Yedidia, W. T. Freeman, and Y.Weiss, "Constructing free energy approximations and generalized belief propagation algorithms," *IEEE Trans. Inform. Theory*, vol. 51, pp. 2282–2312, July 2005.

[29] M. Chertkov and V. Y. Chernyak, "Loop calculus helps to improve belief propagation and linear programming decodings of low-density-parity-check codes," in *Proc. 44th Annual Allerton Conference on Communications, Control and Computing*, Monticello, IL, USA, Sept. 2006. [Online]. Available: http://arxiv.org/abs/cs/0609154.

[30] Y. Lu, C. Measson, and A. Montanari, "TP decoding," in *Proc. 45th Annual Allerton Conference on Communications, Control and Computing*, Monticello, IL, USA, Sept. 2007. [Online]. Available: http://arxiv.org/abs/0710.0564.