Trapping Set Ontology

Bane Vasić, Shashi Kiran Chilappagari, Dung Viet Nguyen and Shiva Kumar Planjery
Department of Electrical and Computer Engineering
University of Arizona
Tucson, AZ 85721, USA
Email: {vasic,shashic,nguyendv,shivap}@ece.arizona.edu

***Abstract*— The failures of iterative decoders for low-density parity-check (LDPC) codes on the additive white Gaussian noise channel (AWGNC) and the binary symmetric channel (BSC) can be understood in terms of combinatorial objects known as trapping sets. In this paper, we derive a systematic method to identify the most relevant trapping sets for decoding over the BSC in the error floor region. We elaborate on the notion of the critical number of a trapping set and derive a classification of trapping sets. We then develop the trapping set ontology, a database of trapping sets that summarizes the topological relations among trapping sets. We elucidate the usefulness of the trapping set ontology in predicting the error floor as well as in designing better codes.**

Keywords: Coding theory; iterative coding techniques

I. Introduction

LDPC codes, invented by Gallager in 1960s [1], have been attracting a large amount of research efforts in the past few years, owing to their capacity-approaching performance under low complexity iterative decoding algorithms. While the behavior of these algorithms is well understood in the limit of the block length tending to infinity (see [2]), many aspects of the finite length analysis are still poorly understood. One such open problem is the phenomenon of the error floor, which can be described as the abrupt degradation in the frame error rate (FER) performance in the high signal-to-noise ratio (SNR) region. It is now known that the error floor arises due to the presence of certain structures, known as stopping sets, trapping sets, near codewords and pseudo-codewords (depending on the channel and the decoding algorithm) that lead to decoder failures (see [3] for an overview of finite length analysis of LDPC codes under different decoding algorithms).

Although the characterization of failures is well understood for the binary erasure channel (BEC) in the form of stopping sets [4], it has been only partially understood for other channels such as the BSC and the AWGNC. Richardson in [5] introduced the notion of trapping sets to characterize failures of decoders and provided a semi-analytical method to estimate the error floors of LDPC codes for transmission over the AWGNC. He gave the necessary and sufficient conditions for a subgraph to form a trapping set for serial flipping decoder for a $(3, 6)$ regular LDPC code and also observed that the trapping sets for the serial flipping algorithm are also trapping sets for the Gallager A and the Gallager B algorithms. Chilappagari *et al.* [6] in the same spirit, presented results on error floor estimation for the BSC with the assumption that the Gallager B algorithm is employed in the decoder. The major difference between these two contributions is that by assuming a hard decision decoding algorithm, issues related to the implementation nuances of the decoding algorithm (such as numerical precision) that can drastically affect the failures of the decoder, are avoided.

It was observed by Chilappagari *et al.* in [3] that the decoding failures for various algorithms and channels are closely related and share a common underlying topological structure. These structures are either trapping sets for iterative decoding algorithms on the BSC or larger subgraphs containing these trapping sets. In this paper, we present an exposition of the methodology of identifying the trapping sets as well as organizing them in a database on the basis of the relations between the topological structures. This database known as the *trapping set ontology* will not only make the identification of trapping sets independent of the code but also aid in the enumeration of trapping sets and guide code construction techniques.

The rest of the paper is organized as follows. In Section II, we provide the preliminaries and background related to LDPC codes. In Section III, we provide definitions related trapping sets and elaborate on the notion of critical number. In Section IV, we describe in detail the concept of trapping set ontology and the method to derive it. In Section V, we explain how trapping set ontology may be used to evaluate the harmfulness of a trapping set. In Section VI, we elucidate on the usefulness of trapping set ontology and its possible applications. Finally, we conclude the paper in Section VII.

II. Preliminaries

A. LDPC Codes

The Tanner graph [7] representation of an LDPC code $\mathcal{C}$, is a bipartite graph with two sets of nodes: n variable (bit) nodes $V = \{1, 2, \ldots, n\}$ and m check (constraint) nodes $C = \{1, 2, \ldots, m\}$. The check nodes (variable nodes resp.) connected to a variable node (check node resp.) are referred to as its neighbors. A vector $\mathbf{w} = (w_1, w_2 \ldots, w_n)$ is a codeword if and only if all the check nodes are satisfied. The support of $\mathbf{w}$, denoted as $\text{supp}(\mathbf{w})$, is defined as the set of all variable nodes (bits) $v \in V$ such that $w_v \neq 0$. The

This work is funded by the NSF under grants CCF-0634969, IHCS-0725405 and CCF-0830245.

bi-adjacency matrix of G gives H, a parity-check matrix of $\mathcal{C}$. The degree of a node is the number of its neighbors. A d_v-left-regular LDPC code has a Tanner graph G in which all variable nodes have degree d_v. Similarly, a d_c-right-regular LDPC code has a Tanner graph G in which all check nodes have degree d_c. A (d_v, d_c) regular LDPC code has a Tanner graph which is d_v-left-regular and d_c-right-regular. This code has rate $r \geq 1 - d_v/d_c$ [1]. The degree of a variable node (check node resp.) is also referred to as the left degree (right degree resp.) or the column weight (row weight resp.). The length of the shortest cycle in the Tanner graph G is called the girth g of G.

B. Channel and the all-zero codeword assumption

In this paper, we consider transmission over the BSC. A variable node is said to be correct if its received value is equal to its original value and corrupt otherwise. To characterize the performance of a coding/decoding scheme for linear codes over any output symmetric channel, one can assume, without loss of generality, the transmission of the all-zero-codeword, i.e. $\mathbf{y} = 0$, when the decoding algorithm satisfies certain symmetry conditions (see Definition 1 and Lemma 1 in [2]). The iterative decoding algorithms that we consider in this paper satisfy these symmetry conditions and henceforth we assume the transmission of the all-zero-codeword. With this assumption, a variable node is correct if it is 0 and corrupt if it is 1.

C. Decoding algorithms

LDPC codes can be decoded with low complexity iterative algorithms. These include the class of message passing algorithms such as the Gallager A/B algorithm and the belief propagation (or sum-product) algorithm (see [2] for various message passing algorithms). Message passing decoders operate by passing messages along the edges of the Tanner graph representation of the code. Every round of message passing (iteration) starts with sending messages from variable nodes (first half of the iteration) and ends by sending messages from check nodes to variable nodes (second half of the iteration). The outgoing message on an edge e is a function of all the incoming messages (and possibly the received value in the case of messages from variable nodes to the check nodes) except the message received on e. There are algorithms which are iterative but do not belong to the class of message passing algorithms, such as the bit flipping (serial or parallel) algorithm. In every iteration of the bit flipping algorithm, constraints (check nodes) are re-evaluated, then variable nodes which are involved in more unsatisfied constraints than satisfied constraints are flipped.

Consider an iterative decoder on the BSC. Let $\mathbf{y} = (y_1, y_2 \ldots, y_n)$ be the input to the decoder and let $\mathbf{x}^l = (x_1^l, x_2^l, \ldots, x_n^l)$, $l \leq D$ be the output vector at the l^{th} iteration. The decoder runs until a valid codeword is found or the maximum number of iterations D is reached, whichever is earlier. The output of the decoder is either a codeword or $\mathbf{x}^D$.

III. Trapping Sets

In this section, we characterize the failures of iterative decoding algorithms using the notion of trapping sets and other related objects. We adopt the definitions of the terms eventually correct variable nodes, trapping sets from [5] and the definitions of inducing sets and fixed sets from [8] and the definition of critical number from [6]. The various aspects of trapping sets discussed in this section as well as Section VI-B and Section VI-C have appeared previously in parts in [6], [8], [9], [10], [11].

A. Definitions

A variable node v is said to be eventually correct if there exists a positive integer q such that for all $l \geq q$, $v \notin \text{supp}(\mathbf{x}^l)$. A decoder failure is said to have occurred if there does not exist $l \leq D$ such that $\text{supp}(\mathbf{x}^l) = \emptyset$.

Definition 1: Let $\mathbf{T}(\mathbf{y})$ denote the set of variable nodes that are not eventually correct. If $\mathbf{T}(\mathbf{y}) \neq \emptyset$, let $a = |\mathbf{T}(\mathbf{y})|$ and b be the number of odd degree check nodes in the sub-graph induced by $\mathbf{T}(\mathbf{y})$. We say $\mathbf{T}(\mathbf{y})$ is an (a, b) trapping set.

The Tanner graph representation of an (a, b) trapping set $\mathcal{T}$ is the sub-graph induced by the variable nodes in the trapping set. This sub-graph consists of a variable nodes in $\mathcal{T}$, represented by $\bullet$, and the check nodes neighboring these variable nodes. We use $\blacksquare$ to represent odd degree check nodes and $\square$ to represent even degree check nodes.

Remark: For each failure of the iterative decoder, there is a corresponding set of corrupt variable nodes: $\mathbf{F} = \text{supp}(\mathbf{x}^D)$. The set $\mathbf{F}$ is not necessarily a trapping set because it may not contain all the variable nodes that are eventually incorrect.

Definition 2: Let $\mathcal{T}$ be a trapping set. If $\mathbf{T}(\mathbf{y}) = \mathcal{T}$ then $\text{supp}(\mathbf{y})$ is a inducing set of $\mathcal{T}$.

Definition 3: Let $\mathcal{T}$ be a trapping set and let $\mathbf{Y}(\mathcal{T}) = \{\mathbf{y} | \mathbf{T}(\mathbf{y}) = \mathcal{T}\}$. The critical number $m(\mathcal{T})$ of trapping set $\mathcal{T}$ is the minimal number of variable nodes that have to be initially in error for the decoder to end up in the trapping set $\mathcal{T}$, i.e.

$$m(\mathcal{T}) = \min_{\mathbf{Y}(\mathcal{T})} |\text{supp}(\mathbf{y})|$$

Definition 4: For transmission over the BSC, $\mathbf{y}$ is a fixed point of the decoding algorithm if $\text{supp}(\mathbf{y}) = \text{supp}(\mathbf{x}^l)$ for all l.

Definition 5: For transmission over the BSC, if $\mathbf{T}(\mathbf{y})$ is a trapping set and $\mathbf{y}$ is a fixed point, then $\mathbf{T}(\mathbf{y}) = \text{supp}(\mathbf{y})$ is a fixed set.

Necessary and sufficient conditions for a set of variable nodes to form a fixed set for the bit flipping algorithm have been derived in [9] and can be generalized to the Gallager A/B algorithms and are given in the following theorem.

Theorem 1: Let $\mathcal{C}$ be an LDPC code with d_v-left-regular Tanner graph G. Let $\mathcal{T}$ be a set consisting of a variable nodes with induced subgraph $\mathcal{I}$. Let the checks in $\mathcal{I}$ be partitioned into two disjoint subsets; $\mathcal{O}$ consisting of checks with odd degree and $\mathcal{E}$ consisting of checks with even degree. Then $\mathcal{T}$ is a fixed set for the Gallager A/B algorithm as well as for

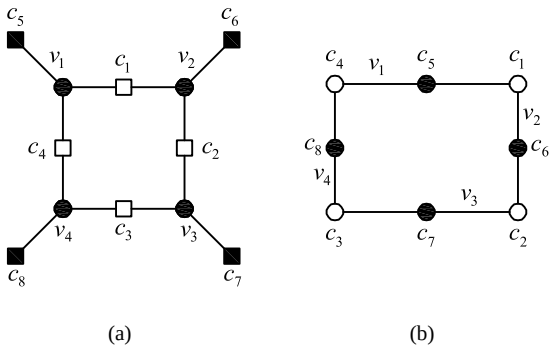

Fig. 1. Graphical representation of $(4, 4)$ trapping set: (a) Tanner graph representation; (b) Line and point representation.

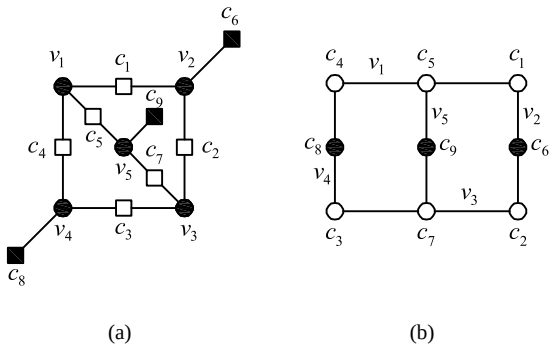

Fig. 2. Graphical representation of $(5, 3)$ trapping set: (a) Tanner graph representation; (b) Line and point representation.

the bit flipping algorithm (serial or parallel) iff : (a) Every variable node in $\mathcal{I}$ has at least $\lceil \frac{d_v}{2} \rceil$ neighbors in $\mathcal{E}$ and (c) No $\lfloor \frac{d_v}{2} \rfloor$ of $\mathcal{O}$ share a neighbor outside $\mathcal{I}$.

B. Example

Fig. 1 shows the sub-graph induced by a $(4, 4)$ trapping set $\mathcal{T} = \{v_1, v_2, v_3, v_4\}$ in a column-weight-three LDPC code. It can be seen that the set of variable nodes in this trapping set are involved in an eight cycle. For the Gallager A/B algorithm, it can be shown that if no two odd degree checks in the induced sub-graph share a variable node outside $\mathcal{T}$, then $\mathcal{T}$ is a fixed set. The critical number of $\mathcal{T}$ is 4, i.e. all the variable nodes v_1, v_2, v_3 and v_4 have to be in error in order for the Gallager A/B decoder to fail in $\mathcal{T}$. For the parallel bit flipping algorithm, it can be shown that $\{v_1, v_3\}$ and $\{v_2, v_4\}$ are inducing sets of $\mathcal{T}$. Therefore the critical number of $\mathcal{T}$ for the parallel bit flipping algorithm is 2. Note that the parallel bit flipping algorithm also fails in $\mathcal{T}$ if all the variable nodes v_1, v_2, v_3 and v_4 are in error and hence $\mathcal{T}$ is also a inducing set. However, it is not a relevant inducing set since its cardinality is much bigger than the cardinality of the other inducing sets.

Fig. 2 shows the graphical representation of a $(5, 3)$ trapping set which is a union of three 8-cycles. This trapping set is a fixed set under the Gallager A/B decoding algorithm because if v_1, v_2, v_3, v_4, v_5 are initially incorrect then the set of incorrect variable nodes is $\{v_1, v_2, v_3, v_4, v_5\}$ after every iteration of the decoder. However, $\{v_1, v_2, v_3, v_4, v_5\}$ is not the smallest inducing set. For the Gallager A/B algorithm, the set of variable node $\{v_2, v_4, v_5\}$ is the smallest inducing set. Hence this trapping set has 3 as its critical number.

IV. Trapping set ontology

Equipped with the description of the conditions for a set of variable nodes to be a trapping set, the next step is to generate a list of subgraphs that satisfy these conditions and thereby build a database of potential trapping sets. The resulting database can then be organized as a trapping set ontology that contains information about the topological relations between the trapping sets. The trapping set ontology will make the trapping set identification process independent of the code and is also vital in enumerating the number of trapping sets, as the knowledge of the topological relationships among the trapping sets greatly simplifies the enumeration.

The problem of creating a list of possible candidates can be related to the coloring problems in graph theory. In order to establish this relationship, we choose an alternate graphical representation of trapping sets based on the incidence structure of lines and points. In combinatorial mathematics, an incidence structure is a triple (P, L, I) where P is a set of "points", L is a set of "lines" and $I \subseteq P \times L$ is the incidence relation. The elements of I are called flags. If $(p, l) \in I$, we say that point p "lies on" line l. In this representation of trapping sets, which we introduced in [12], variable nodes correspond to lines and check nodes correspond to points. A point is shaded black if it has odd number of lines passing through it otherwise it is shaded white. An (a, b) trapping set is thus an incidence structure with a lines and b black shaded points. The graph coloring problems are well studied in literature, and the techniques from this field can be employed [13].

We illustrate the method by a providing a series of examples of identifying fixed sets for column-weight-three LDPC codes of girth eight. The hierarchy that we choose to depict is the parent-child relationship between trapping sets which is based on the sub-graph isomorphism property. Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be two trapping sets having $\mathcal{I}_1$ and $\mathcal{I}_2$ as the Tanner graph representations, respectively. We say that $\mathcal{T}_1$ is a parent of $\mathcal{T}_2$ (or that $\mathcal{T}_2$ is a child of $\mathcal{T}_1$) if $\mathcal{T}_2$ contains a subset whose induced subgraph isomorphic to $\mathcal{I}_1$.

Note: To avoid confusion between the above graphical representations of trapping sets, it can be noticed that the Tanner graph representation of a trapping set always uses ■ or □ to represent check nodes.

Example 1: Consider the $(4, 4)$ and $(5, 3)$ trapping sets as shown in Fig. 1 and Fig. 2. The $(5, 3)$ trapping set can be obtained by adding one variable node v_5 which connect two odd degree check nodes c_5 and c_7. In the point and line representation, this corresponds to adding one line passing through the two black shaded points which represent c_5 and c_7. Since c_5 and c_7 now have two neighbors, they are shaded white. The variable node v_5 needs one more neighboring check node because the code is of column weight three, hence a degree one check node c_9 is added. In the line

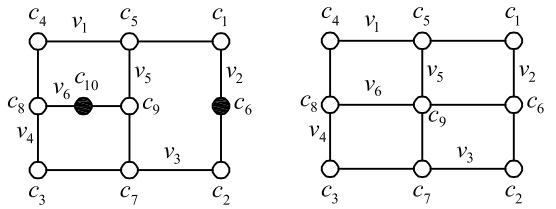

Fig. 3. Children of $(5,3)$ trapping set: (a) $(6,2)$ trapping set; (b) $(6,0)$ trapping set.

and point representation, this corresponds to adding a black shaded point on the line which represents v_5. As a general rule, if a point has an additional line passing through it, the color of its shading will change. This suggests that the additional lines should only pass through the currently black shaded points so that in the resulting sub-graph, all lines have more white shaded points than black shaded points. Also, the girth of the graph has to be preserved. In this example, only line passing through c_5 and c_7 or c_6 and c_8 can be added. If a line is added passing through c_5 and c_6 or c_7 and c_8 then the girth of the graph is reduced to 6.

Example 2: In the same manner, we can add one more variable node to the $(5,3)$ trapping set to get a trapping set of cardinality 6. There are two ways to add such a variable node which yield different results as shown in Fig. 3. If we add a variable node v_6 connecting the check nodes c_8 and c_9 and introduce an odd degree check node c_{10}, then the result is a $(6,2)$ trapping set. On the other hand, if the variable node v_6 is connected to check nodes c_6, c_8 and c_9, then a $(6,0)$ trapping set is obtained. Note that if an LDPC code contains an $(a,0)$ trapping set then it contains codewords of weight a.

Example 3: For illustrative purposes, we again show the line and point representation of the $(4,4)$ trapping set in Fig. 4(a) but with a slightly different way of drawing. It can be seen that the black shaded points which represent check nodes c_5 and c_7 are still on the lines which represent variable nodes v_1 and v_3, respectively. By adding two variable nodes v_5 and v_6 such that v_5 is connected to c_5, v_6 is connected to c_7 and c_9 is the common neighbor of v_5 and v_6, we obtain a $(6,4)$ trapping set as shown in Fig. 4(b). This trapping set is a union of an 8-cycle and a 10-cycle. If we add two variable nodes v_5 and v_6 such that v_5 is connected to c_8, v_6 is connected to c_7 and c_9 is the common neighbor of v_5 and v_6, we also obtain a $(6,4)$ trapping set as shown in Fig. 4(c). However, this $(6,4)$ trapping set is a union of two 8-cycles.

It can be seen that the topological relations among trapping sets follow directly from the way they are identified. A trapping set is a child of another trapping set (its parent) if its incidence structure can be obtained by adding lines to the incidence structure of the other trapping set. Fig. 5 demonstrates the trapping set ontology for column-weight-three LDPC codes under the Gallager A algorithm. Note that only trapping sets up to size six are shown. A more complete list can be found in [14].

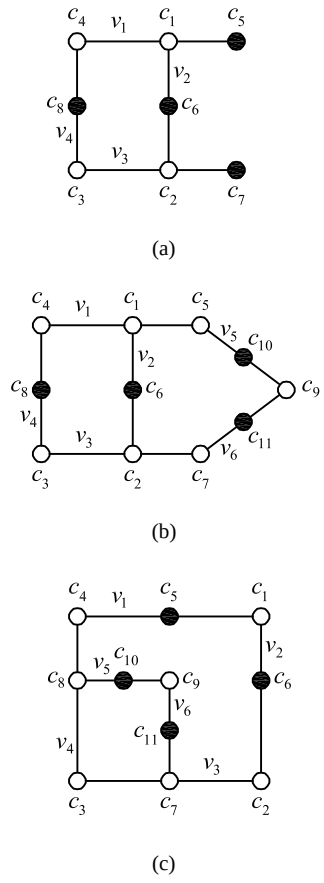

Fig. 4. $(6,4)$ trapping set obtained from $(4,4)$ trapping set: (a) another look of the $(4,4)$ trapping set; (b) $(6,4)$ trapping set formed by a union of an 8-cycle and a 10-cycle; (c) $(6,4)$ trapping set formed by a union of two 8-cycles.

V. Critical Number and the Trapping Set Ontology

The harmfulness of a trapping set $\mathcal{T}$ depends on its critical number $m(\mathcal{T})$ and the number of inducing sets, especially ones with cardinality $m(\mathcal{T})$. The smaller the critical number, the more harmful a trapping set since fewer number of errors can result in decoding failure by ending in that trapping set. Larger number of inducing sets also increases the probability of decoding failure.

In general, if a trapping set $\mathcal{T}_2$ is a child of a trapping set $\mathcal{T}_1$, then $\mathcal{T}_2$ is a more harmful trapping set than $\mathcal{T}_1$. The addition of extra variable nodes to the sub-graph induced by $\mathcal{T}_1$ can either result in a trapping set $\mathcal{T}_2$ with lesser critical number or can introduce new inducing sets that fail on $\mathcal{T}_2$. The $(5,3)$ trapping set of Fig. 1 obtained by the addition of a variable node to the $(4,4)$ trapping set of Fig. 2 has critical number 3 compared to the critical number of 4 for the $(4,4)$ trapping set. Hence, the $(5,3)$ trapping set which is formed by the union of three cycles of length 8 is more harmful than three isolated cycles of length 8.

Now, consider two $(6,4)$ trapping sets shown in Fig. 4(b) and Fig. 4(c). Both are children of the $(4,4)$ trapping set

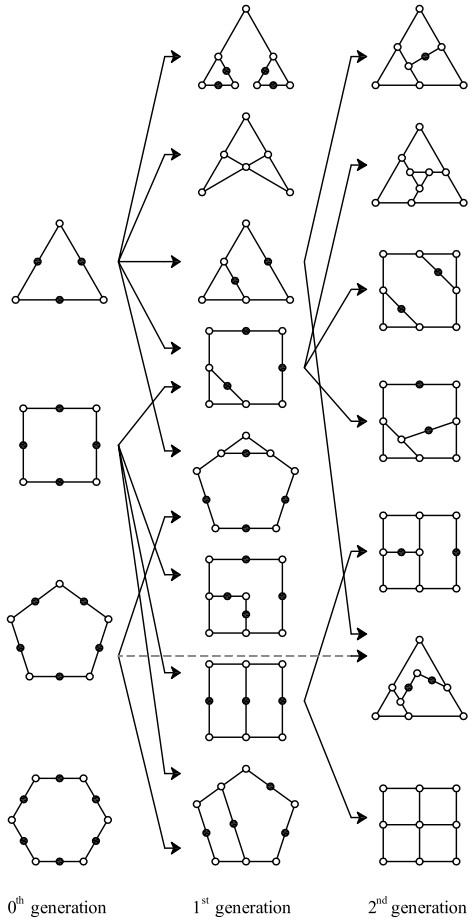

Fig. 5. Trapping set ontology for column weight three codes and Gallager A algorithm.

shown in Fig. 1. Both have critical number $m = 4$. The $(6, 4)$ trapping set which is the union of two 8-cycles (Fig. 4(c)) has an inducing set $\{v_1, v_2, v_5, v_6\}$ of cardinality 4. In the absence of this specific $(6, 4)$ trapping set, the set of variable nodes $\{v_1, v_2, v_5, v_6\}$ will not be an inducing set. In this sense, a $(6, 4)$ trapping set is more harmful than two isolated $(4, 4)$ trapping sets as it has extra inducing sets of cardinality 4.

Similarly, the $(6, 4)$ trapping set which is the union of an 8-cycle and a 10-cycle (Fig. 4(b)) also has an inducing set $\{v_2, v_4, v_5, v_6\}$ of cardinality 4. Again, this $(6, 4)$ trapping set is more harmful than an isolated $(4, 4)$ trapping set and $(5, 4)$ trapping set.

We shall now describe how the trapping set ontology can be used for predicting the error floor behavior of codes as well as in constructing codes that have lower error floors under standard message passing algorithms.

VI. Application of trapping set ontology

A. Enumeration of Trapping Sets

The trapping set ontology can be used to greatly simplify the enumeration of the trapping set classes by exploiting the parent-child relationships between different subgraphs compared to a brute-force search and counting of each individual trapping set class. The first step in the search algorithm would be to count the number of g-cycles in a Tanner graph of girth g since the shortest cycles are the most harmful cycles in the graph. Since the g-cycles can be considered as an (a, a) parent trapping set, we can then search for all the trapping sets that are the children of the g-cycle by simply checking the connections between the odd-degree check nodes of the (a, a) parent. The complexity involved in the search of cycles using a standard tree-based algorithm is linear with the length of the code. Hence, the complexity involved searching for the trapping sets that are children of a parent trapping set will also be linear in length of the code. For example, if we consider a $(3, 6)$ regular girth-8 LDPC code with length of the code being n, the complexity involved in searching for a $(4, 4)$ trapping set using the tree-based algorithm is $(3 \cdot 5^2 \cdot 2^2)n = 300n$. Now in order to search for $(5, 3)$ trapping sets, we can simply take all the $(4, 4)$ parent trapping sets we just found and check if any pair of the degree-one check nodes of the $(4, 4)$ are connected through a variable node. The complexity for the search algorithm will then be twice as much as the complexity involved in searching the $(4, 4)$ trapping sets which is still a linear increase in n. In this manner, by utilizing the trapping set ontology into the search algorithm, the complexity of search is only linear with n.

B. Performance Estimation

The problem of error floor estimation for any given code on a particular channel reduces to identifying the dominant trapping sets (which are typically small subgraphs) for a given decoder and determining the cardinality of such structures present in the Tanner graph of the code. Based on this notion, the work of [5] and [6] provided semi-analytical methods to accurately predict the error floors of codes for the AWGNC and BSC respectively. For the case of the BSC as shown in [6], the trapping sets can be conveniently classified into different classes based on their critical numbers and their strengths in order to simplify the search for relevant trapping sets of a code. The strength of a trapping set is defined as the number of inducing sets with cardinality equal to the critical number that cause the decoder to fail on the trapping set. Using this classification, it becomes sufficient to determine the cardinality of each class of trapping sets present in the Tanner graph of the code and then the contribution of each class towards the error floor estimation for a given SNR can be easily calculated [6]. The cardinality of each trapping set class is determined by searching the graph using techniques outlined in the above section.

Fig. 6 shows the simulated results and estimated error floor for the Margulis code which is a $(3, 6)$ regular LDPC code of length 2640 based on the method described in [6]. It is evident from the result that the prediction of the error floor on the BSC is quite accurate. The interested reader is referred to [6], [15] for details on the estimation method and also for results on a broad class of codes.

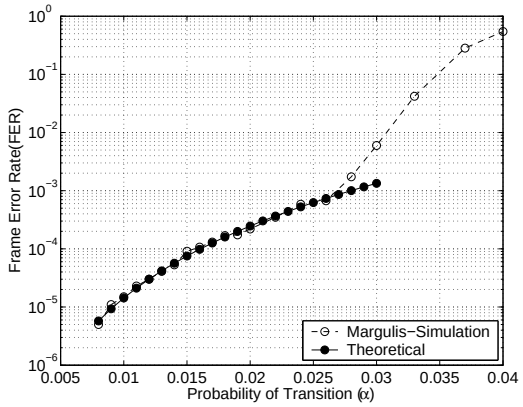

Fig. 6. FER plot for Margulis code under Gallager A algorithm

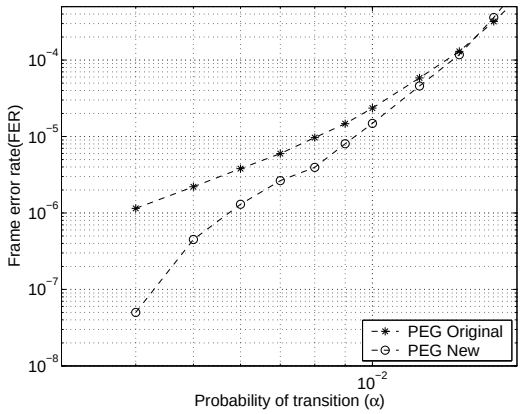

Fig. 7. Example of the FER performance improvement by avoiding trapping sets in code construction

C. Code construction

Given the knowledge of potentially harmful trapping sets for a particular decoder, it is now well understood that an effective code design strategy that significantly improves the slope of the error floor in the FER performance of a code is to avoid these harmful structures while constructing the Tanner graph of the code. For the BSC, the trapping set classification based on the critical number and strength of the trapping set becomes particularly important since the critical number of a trapping set directly affects the slope of the error floor [10]. Hence, by avoiding trapping sets that have low critical number during code construction, we can improve the guaranteed error correction capability of the code.

The parent-child relationship of different subgraphs provided by the trapping set ontology can now be exploited to simplify the construction algorithm of the code. For example, suppose we want to construct a girth-8 code and avoid trapping sets having a critical number of 3 (or lower) for the Gallager-A algorithm on the BSC. From the previous section, we know that the $(5, 3)$ trapping set has a critical number of three and must be avoided. By just avoiding the $(5, 3)$ trapping set, we are already ensuring that the $(6, 2)$, $(6, 0)$ and $(7, 1)$ trapping sets are avoided since they are the children of the $(5, 3)$ trapping set. Hence, in this manner by exploiting the parent-child relationship, the number of trapping sets that need to be considered in the code construction for avoiding is significantly reduced. Also from the construction algorithm's point of view, it becomes much simpler to avoid smaller subgraphs during code construction especially when using standard tree-based construction techniques such as the progressive edge growth (PEG) method [16].

Trapping set ontology can also be helpful in deriving sufficient conditions to guarantee correction of certain number of errors since it can reduce the number of different subgraphs to be analyzed. Based on these premises, the work of [11] provided sufficient conditions on the Tanner graph of the LDPC codes that guarantees correction of three errors. These sufficient conditions are simply the trapping sets that need to be avoided and hence can be directly incorporated into the construction algorithm. In [11], column-weight three

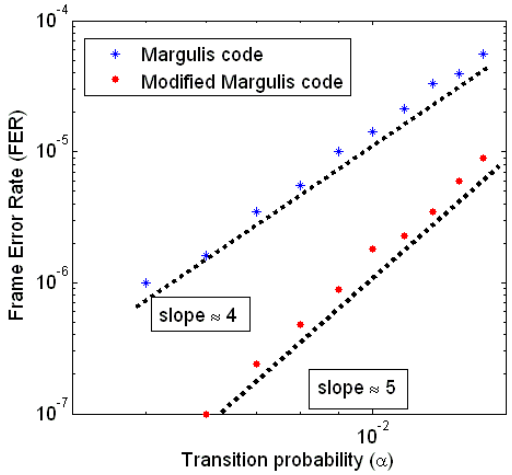

Fig. 8. Example of the FER performance improvement by eliminating trapping sets by using Tanner graph covers

codes were constructed by modifying the PEG algorithm to avoid these structures and the slope of the FER performance of the resulting code was improved from 3 to 4. The performance results of the original code and modified code are shown in Fig. 7. Other strategies apart from modifying the PEG algorithm that can be used to avoid trapping sets during construction are the use of graph covers [10]. In this technique, the method consists of taking two (or more) copies of the same code and swapping edges between the code copies in such a way that the most dominant trapping sets are broken. Performance results comparing the original code with the modified code are shown in Fig. 8. (see [10] for more details on this technique).

VII. Conclusion

In this paper, we have shown a systematic method to identify the most relevant trapping sets for decoding over the BSC in the error floor region. We showed how to develop trapping set ontology, a database of trapping sets that summarizes the topological relations among trapping sets and illustrated the hierarchy for column-weight-three

LDPC codes under the Gallager A algorithm. Examples have been given to show the usefulness of trapping set ontology. Future works include investigating properties of trapping sets and developing trapping set ontology for different decoding algorithms on various channels.

REFERENCES

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

[2] T. J. Richardson and R. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001.

[3] S. K. Chilappagari, M. Chertkov, M. G. Stepanov, and B. Vasic, "Instanton-based techniques for analysis and reduction of error floors of LDPC codes," *IEEE JSAC on Capacity Approaching Codes*, vol. 27, no. 6, pp. 855–865, Aug. 2009.

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

[5] T. J. Richardson, "Error floors of ldpc codes," in *Proc. 41st Annual Allerton Conf. on Communications, Control and Computing*, 2003, pp. 1426–1435. [Online]. Available: http://www.hpl.hp.com/personal/Pascal_Vontobel/pseudocodewords/papers

[6] S. K. Chilappagari, S. Sankaranarayanan, and B. Vasic, "Error floors of LDPC codes on the binary symmetric channel," in *Proc. Int. Conf. on Communications*, vol. 3, 2006, pp. 1089–1094.

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

[8] S. Chilappagari and B. Vasic, "Error-correction capability of column-weight-three LDPC codes," *IEEE Trans. Inform. Theory*, vol. 55, no. 5, pp. 2055–2061, May 2009.

[9] S. K. Chilappagari, D. V. Nguyen, B. Vasic, and M. W. Marcellin, "On trapping sets and guaranteed error correction capability of LDPC codes and GLDPC codes," May 2008, to appear in *IEEE Trans. Inform. Theory*. [Online]. Available: http://arxiv.org/abs/0805.2427

[10] 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.

[11] S. K. Chilappagari, A. R. Krishnan, and B. Vasic, "LDPC codes which can correct three errors under iterative decoding," in *Proc. of IEEE Information Theory Workshop*, Porto, Portugal, May 5-9, 2008, pp. 406–410.

[12] B. Vasic, S. K. Chilappagari, S. Sankaranarayanan, and R. Radhakrishnan, "Failures of the Gallager B decoder: analysis and applications," in *Proc. 2nd Information Theory and Applications Workshop*. University of California at San Diego, Feb. 2006. [Online]. Available: http://ita.ucsd.edu/workshop/06/papers/160.pdf

[13] P. J. Cameron and J. H. V. Lint, *Graphs, Codes and Designs*. New York, NY, USA: Cambridge University Press, 1980.

[14] "Trapping set ontology." [Online]. Available: http://www.ece.arizona.edu/~vasiclab/Projects/CodingTheory/TrappingSetOntology.html

[15] S. K. Chilappagari, B. Vasic, M. Stepanov, and M. Cherktov, "Analysis of error floor of LDPC codes under LP decoding over the BSC," in *Proc. IEEE International Symposium on Information Theory (ISIT '09)*, Seoul, Korea, July 2009, pp. 379–383.

[16] X. Y. Hu, E. Eleftheriou, and D. M. Arnold, "Regular and irregular progressive edge-growth tanner graphs," *IEEE Trans. on Inform. Theory*, vol. 51, no. 1, pp. 386–398, 2005.