Trapping Set Ontology

S. K. Chilappagari, *Member, IEEE,* S. K. Planjery, *Student Member, IEEE,* D. V. Nguyen *Student Member, IEEE,* and Bane Vasic, *Senior Member, IEEE*

***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.**

Index Terms

Low-density parity-check codes, trapping set, iterative decoding algorithm

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 most of the asymptotic behaviors, i.e. behaviors as the block length tends to infinity, of LDPC codes are understood, little is known about their finite length behaviors. One important behavior of finite length LDPC codes is the *error floor* phenomenon. This phenomenon can be described as the abrupt degradation of the frame error rate (FER) performance of the codes in the high signal-to-noise ratio (SNR) region. It is broadly known that the causes of error floor are due to the presence of certain small structures in the Tanner graph of the code that cause the decoder to fail for error patterns of low weight, generally much lower than the error correction capability of the codes under maximum-liklihood decoding. Richardson in [2] introduced the notion of *trapping set* in order to characterize failures of decoders. Using this notion, he provided a semi-analytical method to compute error floors of LDPC codes and presented results for additive white Gaussian noise (AWGN) channel. Chilappagari *et al.* [3] in the same spirit, presented results on error floor estimation for the binary symmetric channel (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. They were also able to associate some trapping sets with certain subgraphs of the Tanner graphs which represent the codes, and hence accurate performance estimation in the error floor region could be made if the number of trapping sets present in the Tanner graph are known. Moreover, graphical representation of trapping sets allows one to study the impact of code's structure on the error floor of the code, consequently giving raise to numerous methods of avoiding trapping sets in code construction and minimizing the effects of trapping sets in the iterative decoders on the BSC [see [4] for examples].

Although the characterization of failures is well understood for the binary erasure channel (BEC) in the form of *stopping sets* which are subgraphs with a deterministic property, it has been only partially understood for other channels such as BSC and AWGN. In [2], Richardson 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. He also proved that the trapping sets for serial flipping algorithm are also trapping sets for the Gallager A and the Gallager B algorithm. In [5], Chilappagari gave the necessary and sufficient conditions for a subgraph to form a *fixed set*, a special case of trapping set, under the Gallager A/B algorithm and the bit flipping algorithm. Using the definition of a *fixed set* for a trapping set, trapping sets can be classified based on their relative harmfulness on the BSC channel using the notion of *critical number*. However, deterministic properties of trapping sets in general are not completely understood, and the problem of identifying and enumerating trapping sets still relies on analyzing all failures of the decoders and explicit graph searching techniques.

It was observed by Chilappagari *et al.* in [6] that by comparing decoding failures of several decoding algorithms on different channels, the decoding failures for various algorithms are closely related and are dependent on only a few topological structures. These structures are either trapping sets for iterative decoding algorithms on the BSC or larger subgraphs containing these trapping sets. Upon analyzing the failures of iterative decoders caused by trapping sets on the BSC, it can be seen that generally trapping sets are subgraphs formed by cycles or union of cycles. These imply that there exists a topological interrelation among trapping sets and in a broader sense, a topological interrelation among error patterns that cause decoding failures for various algorithms on different channels. In this paper, we attempt to find this interrelation and construct a hierarchy of trapping sets, which we call *trapping set ontology*.

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

Manuscript received September 24, 2009. This work is funded by [some rich guys]

S. K. Chilappagari, S. K. Planjery, D. V. Nguyen, and B. Vasic are with the Department of Electrical and Computer Engineering, University of Arizona, Tucson, Arizona, 85721 USA. (emails: {shashic, shivap, nguyendv, vasic}@ece.arizona.edu.

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 elucidate on the usefulness of trapping set ontology and its possible applications. Finally, we conclude the paper in VI.

II. Preliminaries

A. LDPC Codes

A binary LDPC code $\mathcal{C}$ is a linear block code which maps a message block of k information bits to a binary n-tuple. A graphical representation $G = (V \cup C, E)$ of $\mathcal{C}$, also called the Tanner graph [7], is a bipartite graph with two sets of nodes: n variable (bit) nodes $V = \{1, 2, \ldots, n\}$ and m check (constraint) nodes $C = \{1, 2, \ldots, m\}$; and a set of edges E. 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 supp($\mathbf{w}$), is defined as the set of all variable nodes (bits) $v \in V$ such that $w_v \neq 0$. The adjacency matrix of G gives H, a parity-check matrix of $\mathcal{C}$. The degree $d(u)$ of a node $u \in V \cup C$ is the number of its neighbors. A γ-left-regular LDPC code $\mathcal{C}$ has a Tanner graph G in which all variable nodes have degree γ. Similarly, a ρ-right-regular LDPC code $\mathcal{C}$ has a Tanner graph G in which all check nodes have degree ρ. A (γ, ρ) regular LDPC code $\mathcal{C}$ is an LDPC code which is γ-left-regular and ρ-right-regular. This code has rate $r > 1-\gamma/\rho$ [7]. 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 binary symmetric channel (BSC). We define *"flipping"* as the event of a bit changing its value from "0" to "1" or vice versa. The BSC with transition probability p flips a transmitted bit with probability p. A variable node is said to be correct if its received value is equal to its original value and corrupt otherwise.

Since the BSC is a binary-input symmetric-output channel, we can make the all-zero codeword assumption which validity was proved by Richarson and Urbanke in [8]. This assumption relies on the property of standard binary LDPC decoders that the probability of decoding error is equal for any transmitted 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. 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 run 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

A. Definitions

As mentioned above, we assume that the all-zero codeword is transmitted. Consider an LDPC code of length n. A variable node v is said to be *eventually correct* if there exists a positive integer q such that for all $l \geq q$, $v \notin$ supp($\mathbf{x}^l$). A decoder failure is said to have occurred if there does not exist $l \leq D$ such that 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.

Remark: For each failure of the iterative decoder, there is a corresponding set of corrupt variable nodes: $\mathbf{F} =$ 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, such as variable nodes that oscillate between the right value and the wrong value.

Definition 2: Let $\mathcal{T}$ be a trapping set. If $\mathbf{T}(\mathbf{y}) = \mathcal{T}$ then 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 a BSC, $\mathbf{y}$ is a fixed point of the decoding algorithm if supp($\mathbf{y}$) $=$ supp($\mathbf{x}^l$) for all l.

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

Necessary and sufficient conditions for a set of variable nodes to form a fixed set for the Gallager A/B algorithm and for the bit flipping algorithm has been derived in [5] and are given in the following theorem.

Theorem 1: Let $\mathcal{C}$ be an LDPC code with γ-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 the bit flipping algorithm (serial or parallel) iff : (a) Every variable node in $\mathcal{I}$ has at least $\lceil \frac{\gamma}{2} \rceil$ neighbors in $\mathcal{E}$ and (c) No $\lfloor \frac{\gamma}{2} \rfloor$ of $\mathcal{O}$ share a neighbor outside $\mathcal{I}$.

B. Graphical representation

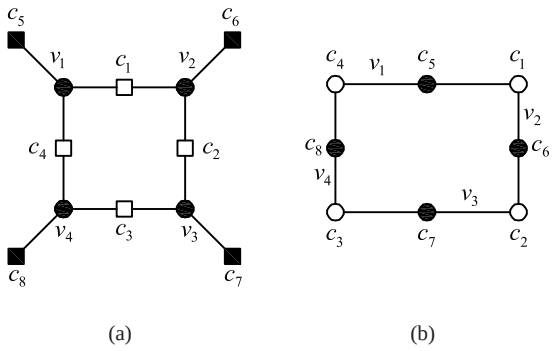
1) Tanner graph representation: The Tanner graph representation of an (a, b) trapping set $\mathcal{T}$ is a sub-graph of the Tanner graph that represents the code. This sub-graph consists of a variable nodes in $\mathcal{T}$, represented by $\bullet$, and the neighboring check nodes to this variable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

2) Line and point representation: Theorem 1 suggests that the problem of creating a list of possible candidates for fixed sets and in general trapping sets 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 $I \in P \times L$ 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, 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 points. The line and point representation also reveals the length of the smallest cycle of the subgraph. It is the smallest number g such that there is a set of g lines and a set of g points in which every point lies on exactly two lines.

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 contain $\blacksquare$ and $\square$ to represents check nodes.

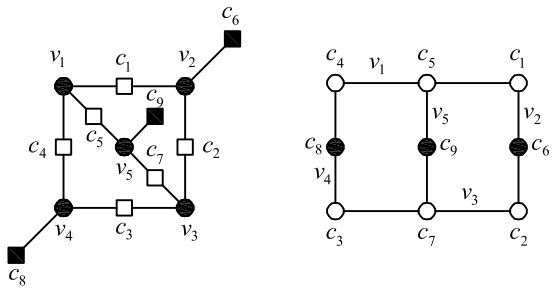
C. Example

Figure 1 shows the Tanner graph representation and the line and point representation of a $(4, 4)$ trapping set $\mathcal{T} = \{v_1, v_2, v_3, v_4\}$ of column weight three LDPC codes. It can be seen that this trapping set forms an 8-cycle. For Gallager A/B algorithm, the only inducing set of $\mathcal{T}$ is $\mathcal{T}$ itself, hence $\mathcal{T}$ 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.



(a) (b)

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



(a) (b)

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

However, it is not a relevant inducing set since its cardinality is much bigger than the cardinality of the other inducing sets.

Figure 2 shows the graphical representation of a $(5, 3)$ trapping set which is a union of two 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

A. Identification of trapping sets and their topological relations

As mentioned above, since trapping sets are either cycles or union of cycles, there exist topological relations among them. Consider an LDPC code whose Tanner graph has girth g. Then as mentioned previously for the case of the BSC, the most harmful trapping sets of the code are typically the shortest cycles which are g-cycles or subgraphs containing unions of g-cycles. For such trapping sets, we say that the subgraph induced by the trapping set has the g-cycle as its *parent* and the subgraph is a *child* of the g-cycle. In general, we say that a subgraph induced by the set of variable node V_p is a parent of a subgraph induced by the set of variable nodes V_c if there exists subset $V'_p \in V_c$ such that the subgraph induced by V'_p

is isomorphic to the subgraph induced by V_p. For simplicity, from now on we shall say that a trapping set is a parent or a child and by that we means the subgraph induced by the trapping set.

Theorem 1 gives an easy way to realize which subgraph is a fixed set for the Gallager A/B algorithm and the bit flipping algorithm. In the line and point representation of a subgraph, if every line has more white points than black points then the subgraph represents a fixed set. This observation leads to a systematic method of identifying fixed sets from its parent. The method is simply described as adding additional lines to a parent subgraph along with changing the color of the points accordingly such that the resulting subgraph satisfies the nescessary and sufficient conditions dictated by Theorem 1. It can be seen that the method is general and can be applied to general trapping sets, as long as the properties of those trapping sets are understood. We choose to illustrate the method by a providing a series of examples of identifying fixed sets for column weight three LDPC codes of girth 8. It will be illustrated in the next subsection that many of these fixed sets contain proper subsets that are inducing sets, hence these fixed sets are trapping sets with critical number smaller than the cardinality of the fixed sets. Henceforth, for the sake of generality, we shall refer to fixed sets as trapping sets.

Example 1: Consider the $(4,4)$ and $(5,3)$ trapping sets as shown in Figure 1 and 2. The $(5,3)$ trapping set can be obtained by adding one variable node v_5 which connect two odd degree check node c_5 and c_7. In the point and line representation, this corresponds to adding one line passing through the two black points which represent c_5 and c_7. Since c_5 and c_7 now have two neighbors, its color changes to white. The variable node v_5 needs one more neighboring check node because the code is of column weight three, hence an odd degree check node c_9 is added. In the line and point representation, this corresponds to adding a black node on the newly added line which represents v_5. As a general rule, if a point has an additional line passing through it, its color will change. This suggests that the additional lines to be added should only passed through the currently black points so that in the resulting subgraph, all lines have more white points than black 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 v_5 and v_6 or $v7$ and v_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 variable node which yield different results as shown in Figure 3. If we add a variable node v_6 connecting to check node 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 has c_6 as its third check node then a $(6,0)$ trapping set is obtained. Note that if an LDPC code contains a $(a,0)$ 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 Figure 4(a) but with a slightly different way of drawing. It can be seen that the black points which represent check node c_5 and

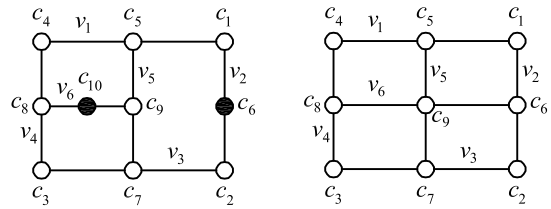


Fig. 3. Children of $(5,3)$ trapping set: (a) $(6,2)$ trapping set; (b) $(6,0)$ trapping set.

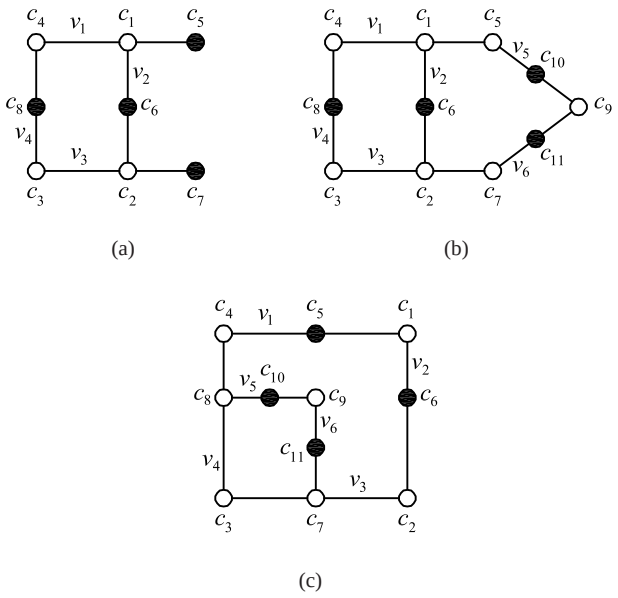


Fig. 4. $(6,4)$ trapping set obtained from $(4,4)$ trapping set: (a) another look of the $(4,4)$ trapping set; (b) $(6,4)$ trapping set formed by a union of an 8-cycle and a 10-cycle; (c) $(6,4)$ trapping set formed by a union of two 8-cycles.

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, which is a union of an 8-cycle and a 10-cycle. This trapping set is shown in Figure 4(b). 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 Figure 4(c). However, this $(6,4)$ trapping set is a union of an 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 induced subgraph can be obtained by adding lines to the induced subgraph of the other trapping set. Figure 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 [9].

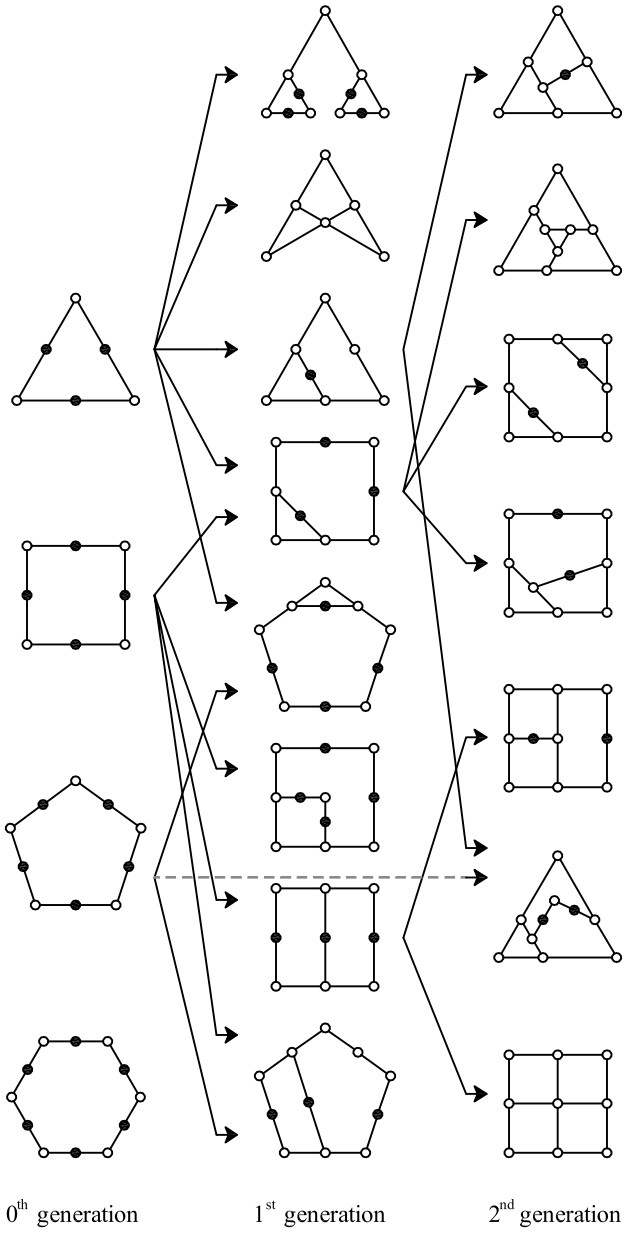


Fig. 5. Trapping set ontology for column weight three codes and Gallager A algorithm.

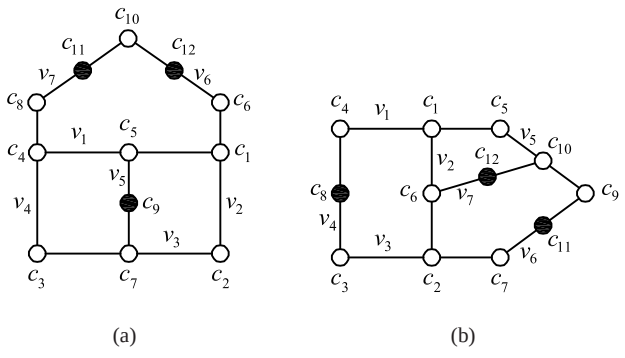


Fig. 6. $(7,3)$ trapping sets: (a) child of $(5,3)$ trapping set; (b) child of $(6,4)$ trapping set.

B. More on critical number of trapping sets

The harmfullness 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.

The topological relations of trapping sets suggest that there is a relation among critical numbers of trapping sets. Based on this, we provide the following conjecture.

Conjecture: The critical number of a trapping set $\mathcal{T}$ is upper bounded by the critical number of its parents.

This conjecture gives a relative measure of the harmfulness of a trapping set based on the harmfulness of its parents as illustrated in the following examples.

Example 4: Consider two $(6,4)$ trapping sets shown in Figure 4(b) and (c). Both are children of the $(4,4)$ trapping set shown in Figure 1. Both has critical number $m=4$. The $(6,4)$ trapping set which is the union of two 8-cycles (Figure 4(c)) has two inducing sets of cardinality 4. Each inducing set consists of variable nodes that form an 8-cycle. On the other hand, the $(6,4)$ trapping set which is the union of an 8-cycle and a 10-cycle (Figure 4(b)) only has one inducing set of cardinality 4. The inducing set, again, consists of the variable nodes which form the 8-cycle. The number of inducing sets indicates that the former trapping set is more harmful than the later. A possible explaination for the difference in the harmfulness of the two trapping sets is that a 10-cycle (or a $(5,5)$ trapping set) is less harmful than an 8-cycle (or a $(4,4)$ trapping set). Thus, the trapping set which is a union of an 8-cycle and a 10-cycle is less harmfull than the trapping set which is a union of two 8-cycles.

Example 5: Consider two $(7,3)$ trapping sets shown in Figure 6. Both are unions of two 8-cycles and one 10-cycle. One trapping set is shown in Figure 6(a) and is a child of the $(5,3)$ trapping set shown in Figure 2. This trapping set has critical number $m=3$. The other trapping set is shown in Figure 6(b). This trapping set is a child of the $(6,4)$ trapping set shown in Figure 4(b) and has critical number $m=4$. It can be seen that the former trapping set is more harmful than the later. The former trapping set also has a more harmful parent, since the $(5,3)$ trapping set has critical number $m=3$ while the $(6,4)$ trapping set has critical number $m=4$.

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

V. Application of trapping set ontology

A. Performance estimation of the iterative decoders in the error floor region

It was previously mentioned that the error floor phenomenon is predominantly caused by the presence of trapping sets in the Tanner graph of a code. Hence, 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 [2] and [3] provided semi-analytical methods to accurately predict the error floors of codes for the AWGN and BSC channels respectively. For the case of the BSC channel as shown in [3], the trapping sets can be conveniently classified into different classes based on their critical numbers and their strengths in order to simply 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 [3]. The cardinality of each trapping set class is determined by searching the graph.

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.

Figure V-A 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 [3]. It is evident from the result that the prediction of the error floor on the BSC is quite accurate.

B. 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

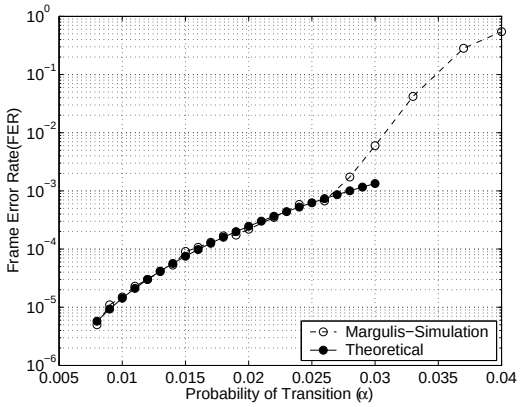


Fig. 7. FER plot for Margulis code under Gallager A algorithm

of the code. For the BSC channel, the trapping set classifcation 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 [4]. 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. Based on the conjecture provided in the previous section that relates the critical numbers of the parent and children, we can see that the relevant trapping sets that need to be avoided are simply the parent trapping sets. 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 channel. 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 progressive edge growth (PEG) [10].

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 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

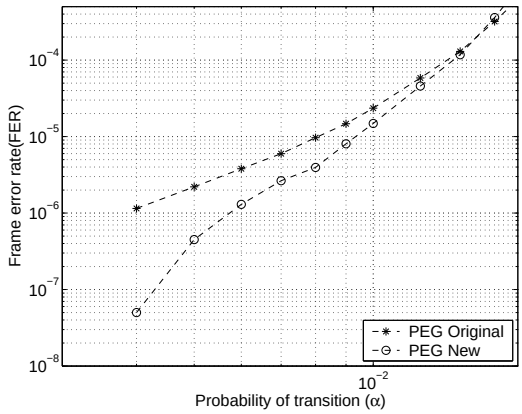


Fig. 8. Example of the FER performance improvement by advoiding trapping sets in code construction

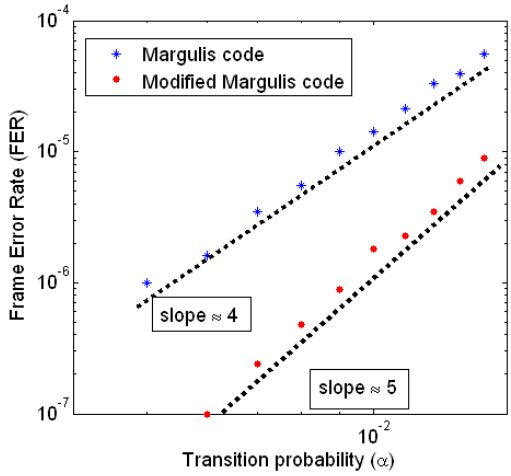


Fig. 9. Example of the FER performance improvement by eliminating trapping sets by using Tanner graph cover

results of the original code and modified code are shown in Figure V-B. Other strategies apart from modifying the PEG algorithm that can be used to avoid trapping sets during construction are the use of graph covers [4]. In this technique, the method it 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 Figure V-B. (see [4] for more details on this technique).

VI. 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 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, "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

[3] 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.

[4] 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. [Online]. Available: http://dx.doi.org/10.1109/TIT.2008.926319

[5] 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.

[6] S. K. Chilappagari, M. Chertkov, M. G. Stepanov, and B. Vasic, "Instanton-based techniques for analysis and reduction of error floors of LDPC codes." [Online]. Available: http://arxiv.org/abs/0903.1624

[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] 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.

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

[10] 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.

[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*, May 5-9, 2008.