When does one redundant parity-check equation matter?

Stefan Laendner[‡], Thorsten Hehn[⋆], Olgica Milenkovic[‡], and Johannes B. Huber[⋆]

[‡] Electrical and Computer Engineering Department
University of Colorado, Boulder, USA
{laendner, milenkov}@colorado.edu

[⋆] Institute for Information Transmission
University of Erlangen-Nuremberg, Erlangen, Germany
{hehn, huber}@LNT.de

***Abstract*— We analyze the effect of redundant parity-check equations on the error-floor performance of low-density parity-check (LDPC) codes used over the additive white Gaussian noise (AWGN) channel. Our findings show that a large number of iterative decoding errors in the $[2640, 1320]$ Margulis code, confined to point trapping sets in the standard Tanner graph, can be corrected if *only one* redundant parity-check equation is added to the decoder's matrix. We also derive an analytic expression relating the number of rows in the parity-check matrix of a code and the parameters of trapping sets in the code's graph.**

Keywords: Margulis Code, Lovàsz Local Lemma, Trapping Sets, Redundant Parity-Checks

I. Introduction

It is well known that the performance of a code depends on the particular form of its parity-check matrix used for iterative decoding. This can be attributed to the fact that the error-rate of a code is reliant on the distribution of combinatorial configurations embedded in its parity-check matrix, including short cycles, stopping sets, and trapping sets [1], [2], [3]. Stopping and trapping sets represent constrained subsets of columns of the parity-check matrix of a code, the size of which can be modified by adding judiciously chosen redundant rows (or equivalently, redundant parity-check equations). Since for transmission over the binary erasure channel (BEC) stopping sets completely characterize the error-floor performance [4], recent work has focused on finding redundant parity-check matrices of codes that maximize the size of the smallest stopping set [5], [6], [7], [8]. Unlike for the case of transmission over the BEC, for signaling over AWGN channels no simple set of combinatorial parameters is known that can accurately specify the performance of an LDPC code. Although information regarding pseudocodewords [6] and trapping sets [3] can be used to approximate the error-floor of a code [3], the task of enumerating such structures within a parity-check matrix is highly non-trivial. Consequently, very little is known about the influence that redundant parity-check equations have on the properties of pseudocodewords and trapping sets and the error-floor performance in general.

We investigate the behavior of the sum-product algorithm in the error-floor region of the Margulis $[2640, 1320]$ code for several different choices of redundant parity-check matrices. We restrict our attention to matrices that can be obtained by augmenting the standard parity-check matrix with *one* redundant row only. This row is chosen in such a way as to increase the cardinality of the set of check nodes with an odd number of connections to a given trapping set. In all cases investigated, one redundant row allows the decoder to recover from otherwise uncorrectable, low-probability error-events. At the same time, it introduces very few short cycles in the code graph and consequently does not compromise the overall performance of the code in the waterfall region. We also propose a simple and efficient method for determining candidate redundant parity-checks that can lead to trapping-error free decoding, based only on the knowledge of unsatisfied check nodes after a given number of decoding iterations. Although the work described in the paper represents a case-study of the Margulis code, most of the described ideas and techniques can be adapted to work for more general classes of codes as well.

The paper is organized as follows. Section II contains the problem setup and introduces some relevant terminology. Section III depicts the implementation of the decoding algorithm used for the analysis. Section IV contains the description of two methods used for identifying redundant parity-check equations that can tailor the structure of a given trapping set to meet certain predefined requirements. In this section we also discuss possible extensions of the methods for a general class of trapping sets and an arbitrary number of redundant parity-check equations. Section V describes a set of analytical results relating the parameters of trapping sets with the number of (possibly redundant) rows in the underlying parity-check matrix of the code.

II. Motivation and Problem Set-Up

The influence of trapping sets (near-codewords) on the performance of LDPC codes in the error-floor regime was first described in [2]; there, it was shown that for the $[2640, 1320]$ Margulis code, most decoding errors are confined to subsets of either 12 or 14 variables for which the induced subgraph contains exactly four unsatisfied checks. This finding can be explained in part by the fact that the presence of *only four* check nodes capable of detecting errors within the given set of variables is not sufficient for correcting all unreliable estimates. Based on the estimates given in [3], 75% of all frame errors occurring at signal-to-noise ratios (measured as $10\log_{10}(E_b/N_0)$) close to 2.4 dB can be attributed to subsets with 12 variables, while approximately 23% of the frame errors arise due to subsets with 14 variables. Furthermore, all subsets with 14 variables were found to represent two-variable extensions of trapping sets with 12 variables.

The aforementioned subsets of variables are usually referred to as *trapping sets*. A formal definition of a trapping set is provided below.

Definition 2.1: For a given $m \times n$ array $U = (u_{i,j})$ with $1 \leq i \leq m$, $1 \leq j \leq n$, the projection of a set of h columns indexed by $j_1, j_2, \ldots, j_h$ is an $m \times h$ sub-array of U consisting of the elements $u_{i,j},\ 1 \leq i \leq m,\ j = j_1, j_2, \ldots, j_h$.

Definition 2.2: Let H be a parity-check matrix of a channel code. An (a, b) trapping set $\mathcal{T}$ is a set of a columns of H with a projection that contains $b > 0$ odd-weight rows. An *elementary* (a, b) trapping set is a trapping set for which all non-zero rows in the projection have either weight one or two, and exactly b rows have weight one.

The results described in [2] and [3] raised a set of interesting questions, some of which will be addressed in this work. The most important questions pertain to the structure of trapping sets that exhibit the strongest influence on the error-floor performance of a code. If, for example, one can increase the number of unsatisfied checks of a trapping set with 12 variables from four to five, is the decoder still going to make an erroneous decision? For a given number of variables, how many unsatisfied checks are needed in order for the decoder to recognize errors confined in a trapping set and to correct them? As will be shown in the next section, both these questions can be addressed within the framework of redundant parity-check equation analysis. The crux of the described study is to show that increasing the number of unsatisfied checks connected to trapping sets of a code[1] in terms of adding redundant parity-check equations can lead to correct decoding of most trapped frames.

Throughout the paper, we will focus on the study of *elementary point trapping sets* only. Trapping sets can be classified into three categories, namely *point*, *periodic*, and *aperiodic* trapping sets [9]. *Point* trapping sets represent subsets of variable nodes that, starting from some iteration, contain all errors ever to occur up to a maximum number of iterations. *Periodic* trapping sets are subsets of the variable nodes that switch from erroneous to error-free values in a periodic fashion. Similarly, *aperiodic* trapping sets represent subsets of variables that are in error at only one stage of the decoding process, and are never revisited during a given number of decoding steps. Most of the $(12, 4)$ and $(14, 4)$ trapping sets identified during decoding of the Margulis code are point trapping sets, and all of them are elementary.

[1]More specifically, the Margulis $[2640, 1320]$ code.

III. Belief-Propagation Algorithm

The error-floor behavior of LDPC codes is, to a certain extent, dependent on the implementation of the sum-product algorithm and the precision of the operations performed. To make our analysis more accessible, we used the publicly available software program from [10] for decoding. This algorithm is briefly summarized below, closely following the notation used in [11].

Denote the set of variable nodes neighboring a check node m by $\mathcal{N}(m) \equiv \{n : H_{mn} = 1\}$. Similarly, denote the set of check nodes neighboring a variable node n by $\mathcal{M}(n) \equiv \{m : H_{mn} = 1\}$; the notation $\mathcal{M}(n)\backslash m$ is reserved for the elements of the set $\mathcal{M}(n)$ excluding the check node m.

During belief propagation, two types of probabilistic messages q_{mn,x_n} and r_{mn,x_n} are exchanged between a check node m and a variable node n if and only if $H_{mn} = 1$. The message q_{mn,x_n} represents the probability that variable n of the transmitted codeword is x_n, given the information provided by all check nodes incident to n, except for m, and the channel output corresponding to variable x_n. The message r_{mn,x_n} is the probability of check node m being satisfied, provided that the variable n has value x_n and all other variables incident to the check node have a separable distribution. The probabilities $q^{(l)}_{mn,x_n}$, where the superscript l indicates the iteration index, are initialized to the prior channel probabilities according to $q^{(0)}_{mn,0} = P(x_n = 0) = p_{n,0}$, and $q^{(0)}_{mn,1} = P(x_n = 1) = p_{n,1} = 1 - p_{n,0}$.

During the *horizontal step* of the l-th iteration of the algorithm, one first computes the probabilities $r^{(l)}_{mn,\,x_n}$ of the observed values z_m arising from $x_n = 0$ or $x_n = 1$, given that the distribution of the variables in $\{x'_n : n' \neq n\}$ is $\{q_{mn',\,0}, q_{mn',\,1}\}$. Equivalently,

$$\begin{aligned} r^{(l)}_{mn,\,0} &= \textstyle\sum_{x_{n'}:n'\in\mathcal{N}(m)\backslash n} P(z_m|x_n = 0,\\ &\quad \{x_{n'} : n' \in \mathcal{N}(m)\backslash n\})\ \textstyle\prod_{n'\in\mathcal{N}(m)\backslash n} q^{(l-1)}_{mn',\,x_{n'}}\\ r^{(l)}_{mn,\,1} &= \textstyle\sum_{x_{n'}:n'\in\mathcal{N}(m)\backslash n} P(z_m|x_n = 1,\\ &\quad \{x_{n'} : n' \in \mathcal{N}(m)\backslash n\})\ \textstyle\prod_{n'\in\mathcal{N}(m)\backslash n} q^{(l-1)}_{mn',\,x_{n'}}. \end{aligned} \tag{1}$$

In the algorithm, the horizontal step is realized in terms of the "fast implementation" approach: The difference of the probabilities, $\delta r_{mn} = r^{(l)}_{mn,\,0} - r^{(l)}_{mn,\,1}$, is computed as

$$\begin{aligned} \delta r_{mn} = (-1)^{z_m} &\textstyle\prod_{n'\in\mathcal{N}(m)} (q^{(l)}_{mn',\,0} - q^{(l)}_{mn',\,1})\\ &/(q^{(l)}_{mn,\,0} - q^{(l)}_{mn,\,1}), \end{aligned} \tag{2}$$

where a denominator value equal to 0 is set to 10^{-20} in order to avoid division by zero. The horizontal step is terminated by computing $r_{mn,\,0} = \frac{1}{2}(1 + \delta_{mn})$ and $r_{mn,\,1} = \frac{1}{2}(1 - \delta_{mn})$.

Since in all steps of the algorithm, the values of $r_{mn,\,0}$ and $r_{mn,\,1}$ appear in denominators, they are lower-bounded by 10^{-20}.

During the *vertical step* of the l-th iteration of standard belief propagation the values of $q_{mn,\,0}$ and $q_{mn,\,1}$ are updated in the following manner. First, the "pseudoposterior probabilities" of x_n being 0 or 1 are computed as

$$\begin{aligned} q_{n,0}^{(l)} &= \alpha_{mn}\, p_{n,0} \prod_{m \in \mathcal{M}(n)} r_{mn,0}^{(l)} \\ q_{n,1}^{(l)} &= \alpha_{mn}\, p_{n,1} \prod_{m \in \mathcal{M}(n)} r_{mn,1}^{(l)}, \end{aligned} \tag{3}$$

where α_{mn} is chosen so that $q_{mn,0} + q_{mn,1} = 1$. These probabilities are used for tentative decoding. The algorithm stops if the hard-decision values of the pseudoposterior probabilities represent a valid codeword. Otherwise, from the pseudoposterior probabilities, the values of $q_{mn,0}$ and $q_{mn,1}$ are calculated according to the equations:

$$\begin{aligned} q_{mn,0}^{(l)} &= \alpha_{mn}\, p_{n,0} \quad q_{n,0}^{(l)}/r_{mn,0}^{(l)} \\ q_{mn,1}^{(l)} &= \alpha_{mn}\, p_{n,1} \quad q_{n,1}^{(l)}/r_{mn,1}^{(l)}. \end{aligned} \tag{4}$$

These probabilities are passed on to the horizontal step of the next iteration.

IV. Trapping Sets and Redundant Parity-Check Equations

Let us assume that during decoding on the standard Tanner graph of the Margulis code, a frame is encountered with errors confined to a $(12, 4)$ trapping set. The problem of interest is to investigate how the estimated values of these 12 trapping variables change with the addition of one redundant parity-check equation. Note that it is tacitly assumed that the channel noise realizations for both experiments are the same. The output of the decoder operating on the augmented Tanner graph depends on the effect of the redundant row on the $(12, 4)$ trapping set. If the projection of the 12 columns on the redundant parity-check equation results in a row of weight 1, the $(12, 4)$ trapping set is converted into an elementary $(12, 5)$ trapping set. Such a set introduces one more unsatisfied check equation, which may help the algorithm in identifying some erroneous variables. Unfortunately, the expansion of a $(12, 4)$ trapping set into a $(12, 5)$ set often does not lead to a correctly decoded frame, but rather "forces" the decoder into a $(14, 4)$ trapping set containing the original $(12, 4)$ trapping set. If, on the other hand, both the projections of the underlying 12 and the extended 14 trapping set variables result in rows of weight one within the redundant parity-check equation, the decoder converges with high probability to the correct codeword.

Figure 1 illustrates the effect that one redundant row in the parity-check matrix of a code can exhibit on the performance of the sum-product algorithm. Figure 1(a) shows the behavior of the iterative decoder when used over the standard Tanner graph of the Margulis code. The decoder first gets trapped in a $(14, 4)$ trapping set, and then settles in its unique $(12, 4)$ trapping subset. Once a trapping event is detected, an appropriately chosen parity-check can be appended to the matrix and the decoding process can be repeated. Figures 1(b) and 1(c) show the performance of the decoder when one redundant row is added to the parity-check matrix in such a way that the projection of the 12 trapped variables onto this row is of

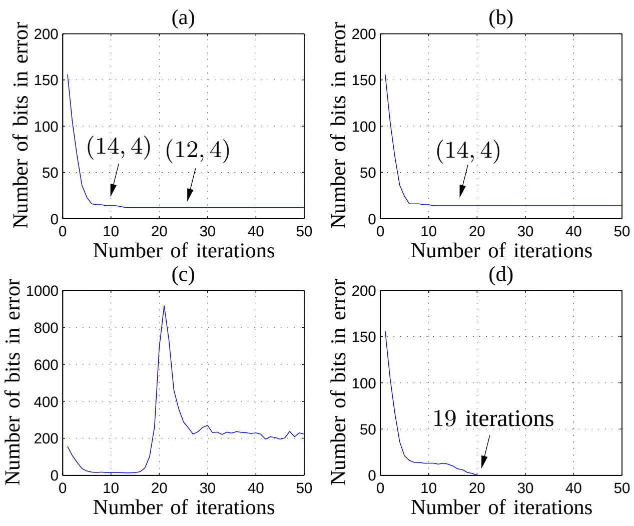

Fig. 1. Number of bits in error for no additional parity-check (a) and different parity-checks appended (b, c, d)

weight one. In Figure 1(b) the decoder gets trapped in the $(14, 4)$ trapping set already observed in Figure 1(a), since the projection of its 14 variables on the redundant row has weight two, and the erroneous variables inside the $(14, 4)$ trapping set are even more biased towards an incorrect decision. The additional parity-check equation used in Figure 1(c) has row-weight one when confined to both the $(12, 4)$ as well as the $(14, 4)$ trapping set. Nevertheless, it does not lead to successful decoding but rather results in the decoder being trapped in an aperiodic trapping set. In Figure 1(d), the output of the decoder for another choice of a redundant parity-check with projection of weight one confined to both the $(12, 4)$ as well as the $(14, 4)$ trapping set is shown. In this case, the decoder successfully identifies the codeword after only 19 iterations.

It is important to observe the following findings of the simulation analysis outlined above. For all $(12, 4)$ and $(14, 4)$ trapping sets investigated, a *single parity-check equation* can be identified that results in correct frame decoding. In all such cases, the projections of both the 12 and the 14 variables contained a vector of weight one within the redundant row. These parity-check equations were found by using two different approaches. One approach, termed genie-aided random search, assumes the knowledge of the erroneous trapping variables. Since such knowledge is not readily available during decoding, another approach, termed structured search, is proposed as well. The structured approach provides for simple and computationally inexpensive identification of appropriate redundant parity checks to be used for decoding trapping errors.

A. Genie-aided random search

This method is primarily used to demonstrate the ease with which a parity-check equation with projections of weight one on both the 12 and 14 trapping set variables can be found. Observe that the existence of such an equation is a consequence of the fact that the Margulis code has minimum distance greater than 14 and that the codewords of its dual code form an orthogonal array of strength at least 14 [12] (we will revisit the properties of a code in terms of its defining orthogonal array in Section V). The genie-aided search algorithm can be summarized as follows:

```
1: List all parity-check equations involving the variable nodes of a $(12, 4)$ trapping set.
2: Construct all linear combinations of the equations found in step 1.
3: Find linear combinations that have weight one when projected to both the $(12, 4)$ and $(14, 4)$ trapping set variables.
4: Append in succession each of these parity-check equations to the standard parity-check matrix and check if the decoder converges to a correct codeword.
```

In all investigated cases, the appended row was not previously contained in the standard parity-check matrix, as can be easily determined based on its weight: while all parity-checks of the original parity-check matrix have weight 6, the weight of the appended rows was found to be an even number in the range from 18 to 50.

B. Structured search

We describe next an approach for identifying "good" redundant parity-checks without referring to the use of side information regarding the position of erroneous variables, but rather exploiting structural properties of the $(12, 4)$ and $(14, 4)$ trapping sets in the regular [2640, 1320] Margulis code.

Throughout the rest of the paper we will call the variables and checks introduced by extending a $(12, 4)$ to a $(14, 4)$ trapping set *expansion variables* and *expansion checks*, respectively. The notation B, E, and O will be used to refer to the basic $(12, 4)$ trapping set, its expansion variables and checks, and the graph outside the $(14, 4)$ trapping set, respectively. Since the Margulis code is regular, with variable node degree $d_v = 3$ and check node degree $d_c = 6$, an elementary (a, b) trapping set with a variables and b check nodes of degree one[2] has a fixed number of checks. If e denotes the number of check nodes connected to two variables within the trapping set, then $e = (3a - b)/2$. Therefore, in order to extend an elementary $(12, 4)$ trapping set to a $(14, 4)$ trapping set, two variables and three checks have to be added.

Since the Margulis code has no four-cycles, at most one expansion check can be connected to both expansion variables. Consequently, either three or four edges emanating from the expansion variables are connected to expansion check nodes, while the remaining edges are connected to check nodes whose degree in the basic $(12, 4)$ trapping set is one. Thus there exist only two possible configurations as shown in Figure 2:

1. Based only on the knowledge of check nodes of degree one within the basic $(12, 4)$ trapping set, it is straightforward to determine the whole expansion set: the two degree-one checks of the basic $(12, 4)$ trapping set are connected through two variable nodes to one additional check node. These two variable nodes are expansion variables.
2. In the second configuration, the expansion variables do not share a check node.

[2]These check nodes, incidentally, correspond to unsatisfied check nodes, that can be readily identified.

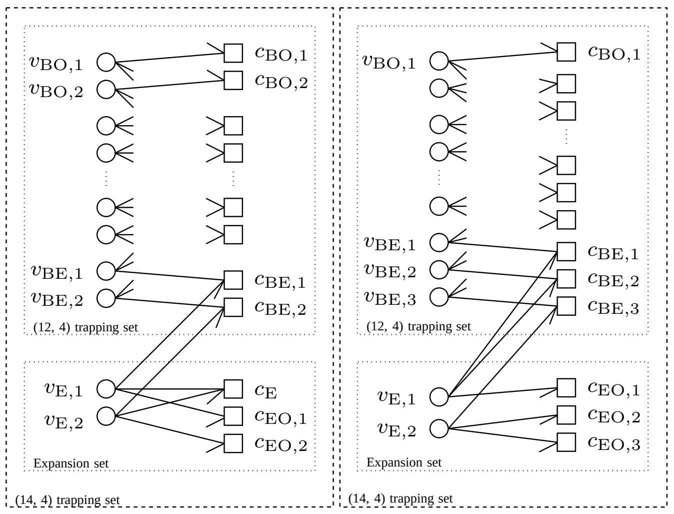

Fig. 2. Trapping set structure of a $(12, 4)$ trapping set and its expansion. (a) configuration 1, (b) configuration 2.

All frames we observed during our simulations correspond to configurations of the first type.

For configuration 1, denote the two expansion variables by $v_{\mathrm{E},1}$ and $v_{\mathrm{E},2}$. Furthermore, denote the check node connected to both these variables by c_{E}. The unsatisfied check nodes in the basic trapping set neighboring the expansion variables are denoted by $c_{\mathrm{BE},1}$ and $c_{\mathrm{BE},2}$, while the variable nodes in the basic trapping set connected to check nodes $c_{\mathrm{BE},1}$ and $c_{\mathrm{BE},2}$ are denoted by $v_{\mathrm{BE},1}$ and $v_{\mathrm{BE},2}$, respectively. The check nodes of degree one in the expansion of the trapping set are $c_{\mathrm{EO},1}$ and $c_{\mathrm{EO},2}$. The two remaining check nodes of degree one in the basic trapping set are termed $c_{\mathrm{BO},1}$ and $c_{\mathrm{BO},2}$, and the variable nodes within the basic $(12, 4)$ trapping set connected to them $v_{\mathrm{BO},1}$ and $v_{\mathrm{BO},2}$, respectively (see Figure 2(a)). The configuration involving all the aforementioned checks and variables is illustrated in Table I.

	Expansion Variables		Basic Trapping Set Variables			
	$v_{\mathrm{E},1}$	$v_{\mathrm{E},2}$	$v_{\mathrm{BE},1}$	$v_{\mathrm{BE},2}$	$v_{\mathrm{BO},1}$	$v_{\mathrm{BO},2}$
c_{E}	1	1	0	0	0	0
$c_{\mathrm{EO},1}$	1	0	0	0	0	0
$c_{\mathrm{EO},2}$	0	1	0	0	0	0
$c_{\mathrm{BE},1}$	1	0	1	0	0	0
$c_{\mathrm{BE},2}$	0	1	0	1	0	0
$c_{\mathrm{BO},1}$	0	0	0	0	1	0
$c_{\mathrm{BO},2}$	0	0	0	0	0	1

TABLE I
PROJECTION OF THE BASIC TRAPPING SET AND EXPANSION VARIABLES

The parity-check equations containing the projections described in Table I can be linearly combined to generate a redundant equation which has a projection of odd weight in both the $(12, 4)$ and the $(14, 4)$ trapping set.

There are three different methods for linearly combining the parity-check equations with projections as shown in Table I.

Method S_1 refers to forming four linear combinations by adding the rows indexed by (c_{E}, $c_{\mathrm{EO},1}$, $c_{\mathrm{BE},2}$), (c_{E}, $c_{\mathrm{EO},2}$, $c_{\mathrm{BE},1}$), ($c_{\mathrm{EO},1}$, $c_{\mathrm{BE},1}$), and ($c_{\mathrm{EO},2}$, $c_{\mathrm{BE},2}$). Observe that all these combinations have a projection of weight one within the basic trapping set. Method S_2 refers to forming two linear combinations by adding the rows indexed by (c_{E}, $c_{\mathrm{EO},1}$) and (c_{E}, $c_{\mathrm{EO},2}$). Observe that all these combinations have a projection of weight one within the projection of the expansion variables. Method S_3 differs from the previous methods in so far that it may generate redundant parity-check equations with odd-weight projections that are not necessarily of weight one. Candidate equations are obtained by linear combinations of the rows indexed by (c_{E}, $c_{\mathrm{BO},1}$), (c_{E}, $c_{\mathrm{BO},2}$), (c_{E}, $c_{\mathrm{BE},1}$, $c_{\mathrm{BE},2}$, $c_{\mathrm{BO},1}$), (c_{E}, $c_{\mathrm{BE},1}$, $c_{\mathrm{BE},2}$, $c_{\mathrm{BO},2}$), ($c_{\mathrm{BO},1}$, $c_{\mathrm{BO},2}$, $c_{\mathrm{EO},1}$, $c_{\mathrm{BE},2}$), and ($c_{\mathrm{BO},1}$, $c_{\mathrm{BO},2}$, $c_{\mathrm{EO},2}$, $c_{\mathrm{BE},1}$). The Hamming weight of the constructed rows is an even integer between 10 and 24. The lower bound 10 is met by the constructions S_1 and S_2 if the supports of two added parity-checks share one element. There exist no further common elements in this case as no cycles of length 4 are present. Method S_3 meets the upper bound 24: four parity-check equations are added and none of the variables listed in Table I occur in the support of more than one parity-check.

If a trapping set of the form shown in Figure 2(b) occurs, the two expansion variables have no common check node. In this case expansion variable $v_{\mathrm{E},1}$ is connected to two of the degree-one check nodes of the basic trapping set.[3] If there is only one variable node connected to two degree-one checks, denoted by $c_{\mathrm{BE},1}$ and $c_{\mathrm{BE},2}$, the variable of interest is the expansion variable $v_{\mathrm{E},1}$, which is also connected to expansion check node $c_{\mathrm{EO},1}$. Observe that expansion variable $v_{\mathrm{E},2}$ is strongly influenced by its two neighboring check nodes connected to the outside graph and can not be uniquely determined. Due to this limited knowledge of the expansion set, there are only two possible ways to generate a redundant parity-check with an odd projection weight on the basic trapping set, involving the sums of the rows ($c_{\mathrm{EO},1}$, $c_{\mathrm{BE},1}$) as well as ($c_{\mathrm{EO},1}$, $c_{\mathrm{BE},2}$). A similar analysis can be conducted for $(14, 4)$ trapping sets. Due to space limitations, details are omitted.

Table II illustrates the application of methods S_1, S_2, and S_3, as well as the genie-aided random search for identifying redundant parity-check equations that lead to correct decoding of 15 frames erroneously decoded on the standard Tanner graph. The table includes data describing the parameters of the sets in which the decoder was initially trapped, the number of parity-check equations that had to be added in order to generate the redundant equation used in the genie-aided random search method, and the number of iterations performed by the decoder until successful decoding. Also listed are the types of structured methods used to generate the redundant parity-check equation. Underlined symbols refer to methods that used the minimum number of iterations to converge to the correct solution. This number is also shown in the rightmost column of the table. Although decoding with an augmented parity-check matrix should only be performed in the error-floor region (by restarting the decoder if it gets trapped), simulation results indicate that the performance in the waterfall region is not influenced by the presence of one redundant parity-check, so that decoding can be performed with the redundant parity-check matrix for all SNR values.

As a final remark, we would like to point out that the location of the non-zero entry within the projection of the redundant parity-check equation on the $(12, 4)$ or $(14, 4)$ trapping set does not seem to influence the performance of the decoder.

V. Redundancy Versus Trapping Set Size: An Analytical Study

As already mentioned in the previous sections, there is no guarantee that one redundant parity-check equation can change the structure of a trapping set sufficiently enough to prevent the decoder from making erroneous decisions on its constituent variables. This problem seems to be especially prominent when the variables in a trapping set have a very small number of check nodes connected to them an odd number of times. It is therefore of interest to investigate the fundamental trade-offs between the number of rows in a parity-check matrix of a code and the "smallest size" of a trapping set from a given class. For this purpose, we will try to connect $T_H(a, b)$, the number of (a, b) trapping sets in a parity-check matrix H, to its number of rows.

Definition 5.1: ([12, p.5]) An orthogonal array of strength t is defined as a matrix of dimensions $m \times n$ such that every $m \times t$ sub-array contains *each* possible t-tuple *the same number of times*. The codewords of an $[n, k, d]$ code $\mathcal{C}$ form an orthogonal array of dimension $2^k \times n$ of strength $d^{\perp} - 1$, where $d^{\perp}$ denotes the dual distance of $\mathcal{C}$.

Proposition 5.1: Let H consist of all $2^{n-k} \geq 2$ codewords of the dual of an $[n, k, d]$ code $\mathcal{C}$. Then $T_H(a, b) = 0$ for all a and b such that $1 \leq a \leq d - 1$, and $b \neq 2^{n-k-1}$.

Proposition 5.1 shows that a parity-check matrix consisting of all codewords of the dual code cannot contain trapping sets with $1 \leq a \leq d - 1$ variables that have fewer than 2^{n-k-1} checks connected to them an odd number of times. On the other hand, it is of much larger importance to determine if there exist parity-check matrices of a code free of trapping sets with parameters (a, s), for $1 \leq s < b$, and some relatively small value b.

Theorem 5.2: Let $\mathcal{C}$ be an $[n, k, d]$ code and $\mathcal{C}^{\perp}$ its dual. Define the $\mathcal{M}_{\mathcal{C}}(m)$ ensemble to be the set of all $m \times n$ matrices with rows chosen uniformly and independently from the set of 2^{n-k} codewords of $\mathcal{C}^{\perp}$. Furthermore, let $1 \leq a \leq \lfloor (d-1)/2 \rfloor$ be fixed and let $\theta(a, b)$ be the number of trapping sets with parameters $((a, s))$, $1 \leq s < b$, in a randomly chosen matrix of $\mathcal{M}_{\mathcal{C}}(m)$. If

$$\mathrm{e}\, a \binom{n-1}{a-1} \left(\frac{1}{2}\right)^m \sum_{j=0}^{b-1} \binom{m}{j} \leq 1, \tag{5}$$

[3] One must keep in mind that the choice for such a variable might not be unique.

Genie-aided random search, IV-A				Structured search, IV-B	
Frame	Type	# of comb. parity-checks	Requ. iterations	Working structured search methods	Requ. iterations
1	$(12,4)$	4	24	S_1, $\underline{S_3}$	16
2	$(12,4)$	9	22	S_1, $\underline{S_2}$	22
4	$(12,4)$	6	24	$\underline{S_3}$	36
5	$(12,4)$	12	28	$\underline{S_3}$	25
6	$(14,4)$	7	27	S_1, $\underline{S_3}$	20
7	$(12,4)$	4	20	$\underline{S_1}$, S_2, S_3	13
8	$(14,4)$	9	18	$\underline{S_3}$	17
9	$(12,4)$	5	20	$\underline{S_1}$	24
10	$(12,4)$	4	22	$\underline{S_3}$	17
11	$(14,4)$	5	18	$\underline{S_3}$	14
12	$(12,4)$	8	21	$\underline{S_1}$	27
13	$(12,4)$	4	19	$\underline{S_1}$, S_2, S_3	16
14	$(12,4)$	5	26	$\underline{S_3}$	16
15	$(14,4)$	6	24	$\underline{S_1}$	26
16	$(12,4)$	5	21	$\underline{S_3}$	16

TABLE II

SUMMARY OF SIMULATION RESULTS FOR PROCESSED FRAMES USING THE GENIE-AIDED RANDOM SEARCH AND STRUCTURED SEARCH METHODS. THE SOFTWARE USED TO GENERATE PART OF THE RESULTS IS AVAILABLE UPON REQUEST FROM THE AUTHORS.

then $P\{T(a,b)=0\} > 0$. Consequently, if m satisfies (5), then there exists a parity-check matrix of $\mathcal{C}$ with no more than $m+n-k-d+1$ rows and $\theta(a,b)=0$.

Proof: The proof of the claimed result is based on Lovàsz local lemma and extensions thereof [13], [14], [15], stated below.

Lemma 5.3: Let $E_1, E_2, \ldots, E_N$ be a set of events in an arbitrary probability space. Suppose that each event E_i is independent of all other events E_j, except for at most τ of them, and that $P\{E_i\} \leq p$ for all $1 \leq i \leq N$. If

$$\mathrm{e}\, p\, (\tau+1) \leq 1, \tag{6}$$

then $P\{\bigcap_{i=1}^{N} \overline{E}_i\} > 0$.

Let E_i be the event that the projection of the i-th collection of a columns from a randomly chosen matrix in $\mathcal{M}_{\mathcal{C}}(m)$ contains fewer than b odd rows. Then

$$P\{E_i\} = \left(\frac{1}{2}\right)^m \sum_{j=0}^{b-1} \binom{m}{j}.$$

This is due to the fact that the codewords of the dual code form an orthogonal array of strength $d-1$, and that consequently, even and odd weight rows in the projection are equally likely. In this setting, $P\{\bigcap \overline{E}_i\}$ denotes the probability that the randomly chosen matrix is free of trapping sets with parameters (a,s), where $1 \leq s < b$. In order to complete the proof, it suffices to observe that the dependence number of the events E_i is upper bounded as

$$\tau + 1 \leq a \binom{n-1}{a-1},$$

since any subset of a columns depends only on other subsets that share at least one column with it, provided that a is upper bounded as stated in the theorem. Additional $n-k-d+1$ rows may be required to make the randomly chosen matrix have rank $n-k$. ∎

ACKNOWLEDGEMENT

This work was in part supported by a German Academic Exchange Service (DAAD) fellowship awarded to Thorsten Hehn, NSF Grant CCF-0514921, and a fellowship by the Institute of Information Transmission (LIT) at the University of Erlangen-Nuremberg, Germany, awarded to Stefan Laendner.

REFERENCES

[1] C. Di, D. Proletti, I. Telatar, T. Richardson, and R. Urbanke, "Finite length analysis of low-density parity-check codes," *IEEE Trans. on Inform. Theory*, vol. 48, pp. 1570–1579, June 2002.

[2] D. MacKay and M. Postol, "Weaknesses of Margulis and Ramanujan-Margulis low-density parity-check codes," *Electronic Notes in Theoretical Computer Science*, vol. 74, 2003.

[3] T. Richardson, "Error-floors of LDPC codes," in *Proceedings of the 41st Annual Allerton Conference on Communication, Control and Computing*, pp. 1426–1435, Sept. 2003.

[4] A. Orlitsky, K. Viswanathan, and J. Zhang, "Stopping set distribution of LDPC code ensembles," *IEEE Trans. on Info. Theory*, vol. 51, pp. 929–953, March 2005.

[5] J. Han and P. Siegel, "Improved upper bounds on stopping redundancy," in *Proc. IEEE Int. Symp. Inform. Theory (ISIT'06)*, (Seattle, Washington), July 2006.

[6] P. Vontobel and R. Koetter, "Graph-cover decoding and finite-length analysis of message-passing iterative decoding of LDPC codes," *submitted to IEEE Trans. on Inform. Theory*, Dec. 2005.

[7] M. Schwartz and A. Vardy, "On the stopping distance and stopping redundancy of codes," *submitted to IEEE Trans. on Inform. Theory*, March 2005.

[8] J. Weber and K. Abdel-Ghaffar, "Stopping and dead-end set enumerators for binary Hamming codes," in *Proceedings of the Twenty-sixth Symposium on Information Theory in the Benelux*, (Brussels, Belgium), pp. 165–172, May 2005.

[9] S. Laendner and O. Milenkovic, "Algorithmic and combinatorial analysis of trapping sets in structured LDPC codes," in *Proceedings of the International Conference on Wireless Networks, Communications, and Mobile Computing (WirelessComm2005)*, (Maui, Hawaii), June 2005.

[10] I. Kozintsev, Website, http://www.kozintsev.net/soft.html

[11] D. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. on Inform. Theory*, vol. 45, pp. 399–431, March 1999.

[12] A. Hedayat, N. Sloane, and J. Stufken, *Orthogonal Arrays: Theory and Applications*. New York: Springer Verlag, 1999.

[13] D. Deng, D. Stinson, and R. Wei, "The Lovász local lemma and its applications to some combinatorial arrays," *Designs, Codes and Cryptography*, vol. 32, pp. 121–134, May 2004.

[14] N. Alon and J. Spencer, "The probabilistic method," in *Interscience Series in Discrete Mathematics and Optimization*, John Wiley, 2000.

[15] P. Carey and A. P. Godbole, "Partial covering arrays and a generalized Erdös-Ko-Rado property," *preprint, available at http://arxiv.org/abs/math.CO/0512139*, Dec. 2005.