Chapter 13
An Exhaustive Tree Search for Stopping Sets of LDPC Codes

13.1 Introduction and Preliminaries

The performance of all error-correcting codes is determined by the minimum Hamming distance between codewords. For codes which are iteratively decoded such as LDPC codes and turbo codes, the performance of the codes for the erasure channel is determined by the stopping set spectrum, the weight (and number) of erasure patterns which cause the iterative decoder to fail to correct all of the erasures. Codes which perform poorly on the erasure channel do not perform well on the AWGN channel. To determine all of the stopping sets of a general (n, k) code is a prohibitive task, for example, a binary (1000, 700) code would require evaluation of 2^{1000} possible stopping sets. It should be noted by the reader that all codewords are also stopping sets, but most stopping sets are not codewords. Fortunately the properties of particular types of codes may be used to reduce considerably the scale of the task, and in particular codes with sparse parity-check matrices such as LDPC codes and turbo codes are amenable to analysis in practice. As the tree search is exhaustive, the emphasis is first on focusing the search so that only low-weight stopping sets are found, up to a specified weight, and second the emphasis is on the efficiency of the algorithms involved.

In a landmark paper in 2007, Rosnes and Ytrehus [7] showed that exhaustive, low-weight stopping set analysis of codes whose parity-check matrix is sparse is feasible using a bounded tree search over the length of the code with no distinction between information and parity bits. A previous paper on the same topic of an exhaustive search of stopping sets of LDPC codes by Wang et al. [2] used a different and much less efficient algorithm. In common with this earlier research, we use similar notation in the following preliminaries.

The code $\mathscr{C}$ is defined to be binary and linear of length n and dimension k and is a k-dimensional subspace of $\{0, 1\}^n$, and may be specified as the null space of a $m \times n$ binary parity-check matrix $\mathbf{H}$ of rank $n - k$. The number of parity-check equations, m of $\mathbf{H}$ satisfies $m \geq (n - k)$, although there are, of course, only $n - k$ independent parity-check equations. It should be noted, as illustrated in the results below, that the

M. Tomlinson et al., *Error-Correction Coding and Decoding*,
Signals and Communication Technology,
DOI 10.1007/978-3-319-51103-0_13

number of parity-check equations m in excess of $n-k$ can have a dramatic effect on the stopping set weight spectrum, excluding codewords of course, as these are not affected.

As in [7], $\mathscr{S}$ is used to denote a subset of $\{0, 1\}^n$, the set of all binary vectors of length n. At any point in the tree search, a constraint set, $\mathscr{F}$ is defined consisting of bit positions p_i and the states of these bit positions $s_{p_i}, s_{p_i} \in \{0, 1\}^n$. The support set $\chi(\mathscr{F})$ of the constraint set, $\mathscr{F}$, is the set of positions where $s_{p_i} = 1$, and the Hamming weight of $\mathscr{F}$ is the number of such positions. The sub-matrix $\mathbf{H}_{\chi(\mathscr{F})}$ consists of all the columns of $\mathbf{H}$ where $s_{p_i} = 1$, and the row weight of $\mathbb{H}_{\chi(\mathscr{F})}$ is the number of $1's$ in that row. An active row of $\mathbf{H}_{\chi(\mathscr{F})}$ is a row with unity row weight. It is obvious that if all rows of $\mathbf{H}_{\chi(\mathscr{F})}$ have even row weight then $\mathscr{F}$ is a codeword, noting that for an iterative decoder codewords are also stopping sets. If at least one row has odd weight, 3 or higher and there are no active rows then $\mathscr{F}$ is a stopping set but not a codeword. If there are active rows then $\mathscr{F}$ has either to be appended with additional bit positions or one or more states s_{p_i} need to be changed to form a stopping set. With this set of basic definitions, tree search algorithms may be described which carry out an exhaustive search of $\{0, 1\}^n$ using a sequence of constraints $\mathscr{F}$ to find all stopping sets whose Hamming weight is $\leq \tau$.

13.2 An Efficient Tree Search Algorithm

At any given point in the search, the *constraint set* F is used to represent the set of searched known bits (up to this point) of a code $\mathscr{C}$, which forms a *branch* of the tree in the tree search. The set of active rows in $\mathbf{H}$ is denoted by $\{\mathfrak{h}_0, ..., \mathfrak{h}_{\phi-1}\}$, where ϕ is the total number of active rows. A constraint set F with size n is said to be *valid* if and only if there exists no active rows in $\mathbf{H}^{(F)}$. In which case the constraint set is equal to a stopping set. The pseudocode of one particularly efficient algorithm to find all the stopping sets including codeword sets with weight equal to or less than τ is given in Algorithm 13.1 below. Each time a stopping set is found, it is stored and the algorithm progresses until the entire 2^n space has been searched.

The modified iterative decoding is carried out on a n-length binary input vector containing erasures in some of the positions. Let $r_j(F)$ be the rank (ones) of row j, $j \in \{0, ..., m-1\}$ for the constrained position $\{p_i : (p_i, 1) \in F\}$ intersected by row j on $\mathbf{H}$. And let $r'_j(F)$ be the rank of row j for the unconstrained position $\{p_i : (p_i, 1) \in \{0, ..., n-1\} \backslash F\}$ intersected by row j on $\mathbf{H}$. The modified iterative decoding algorithm based on belief-propagation decoding algorithm over the binary erasure channel is shown in Algorithm 13.2. As noted in the line with marked (*), the modified iterative decoder is not invoked if the condition of $r_j \leq 1$ and $r'_j = 1$ is not met; or the branch with constraint set F has condition of $r_j = 1$ and $r'_j = 0$. This significantly speeds up the tree search. As noted in the line with marked (*), the modified iterative decoder is not necessary to call, if the condition of $r_j \leq 1$ and

Algorithm 13.1 Tree search based Stopping Set Enumeration (TSSE)

repeat
  Pick one untouched branch as a constraint set F.
  if $|F| = n$ and $w(F) \leq \tau$ **then**
    Constraint set F is saved, if F is valid
  else
    1). Pass F to the modified iterative decoder (*) with erasures in the unconstrained positions.
    2). Construct a new constraint set F' with new decoded positions, which is the extended branch.
    if $|F'| = n$ and $w(F') \leq \tau$ **then**
      Constraint set F' is saved, if F' is valid
    else if No contradiction is found in $\mathbf{H}^{(F')}$, and $w'(F') \leq \tau$ **then**
      a). Pick an unconstrained position p.
      b). Extending branch F' to position p to get new branch $F'' = F' \bigcup \{(p, 1)\}$ and branch $F''' = F' \bigcup \{(p, 0)\}$.
    end if
  end if
until Tree has been fully explored

Algorithm 13.2 Modified Iterative Decoding

Get rank $\mathbf{r}(F)$ and $\mathbf{r}'(F)$ for all the equation rows on $\mathbf{H}$.
repeat
  if $r_j > 1$ **then**
    Row j is flagged
  else if $r_j = 1$ and $r'_j = 0$ **then**
    Contradiction $\rightarrow$ Quit decoder
  else if $r_j \leq 1$ and $r'_j = 1$ **then**
    1). Row j is flagged
    2). The variable bit i is decoded as the **XOR** of the value of r_j.
    3). Update the value of r_j and r'_j, if $H_{ji} = 1$.
  end if
until No new unconstrained bit is decoded

$r'_j = 1$ is not met; or the branch with constraint set F can be ignored, if condition of $r_j = 1$ and $r'_j = 0$ occurs. Thus the computing complexity can be significantly reduced than calling it for every new branch with the corresponding constraint set F.

13.2.1 An Efficient Lower Bound

The tree search along the current branch may be terminated if the weight necessary for additional bits to produce a stopping set plus the weight of the current constraint set F exceeds τ. Instead of actually evaluating these bits, it is more effective to calculate a lower bound on the weight of the additional bits. The bound uses the active rows $\mathscr{J}(F) = \{I_{i_0}(F), ..., I_{i_{q-1}}(F)\}$, where $I_{i_0}(F)$ is the set of active rows with constraint set F corresponding to the i_0th column $\mathbf{h}_{i_0}$ of $\mathbf{H}$, and q is the number

Table 13.1 Low-weight stopping sets and codewords of known codes

Code Name	s_m	N_{s_m}	$N_{s_m}+1$	$N_{s_m}+2$	$N_{s_m}+3$	$N_{s_m}+4$	$N_{s_m}+5$	$N_{s_m}+6$
Tanner (155, 64) [6]	18	465 (0)	2015 (0)	9548 (1023)	23715 (0)	106175 (6200)	359290 (0)	1473585 (43865)
QC LDPC (1024, 512) [4]	15	1 (1)	1 (0)	0 (0)	1 (1)	6 (1)	6 (2)	12 (4)
PEG Reg (256, 128) [3, 5]	11	1 (0)	11 (7)	22 (12)	51 (28)	116 (46)	329 (113)	945 (239)
PEG Reg (504, 252) [3, 5]	19	2 (0)	5 (2)	8 (0)	27 (5)	78 (0)	241 (30)	()
PEG iReg (504, 252) [3, 5]	13	2 (1)	1 (1)	5 (5)	13 (11)	31 (16)	52 (28)	124 (60)
PEG iReg (1008, 504) [3, 5]	13	1 (1)	0 (0)	0 (0)	3 (3)	3 (3)	4 (4)	5 (3)
MacKay (504, 252) [5]	16	1 (0)	3 (0)	3 (0)	12 (0)	36 (2)	106 (0)	320 (22)

Table 13.2 WiMax 1/2 LDPC Codes

i	S_{min}	$N_{s_{min}}$	$N_{s_{min}+1}$	$N_{s_{min}+2}$	$N_{s_{min}+3}$	$N_{s_{min}+4}$	$N_{s_{min}+5}$	$N_{s_{min}+6}$	$N_{s_{min}+7}$	$N_{s_{min}+8}$
0	13	24(24)	0(0)	0(0)	24(24)	0(0)	24(0)	120(72)	312(96)	()
1	18	56(0)	140(56)	56(56)	308(84)	420(168)	756(224)	2296(476)	5460(1288)	()
2	18	32(0)	0(0)	96(64)	128(32)	192(96)	704(352)	992(224)	1888(672)	()
3	19	36(36)	36(36)	144(0)	324(36)	828(180)	810(162)	2304(576)	()	()
4	19	120(80)	120(40)	160(0)	280(160)	400(120)	880(120)	1760(560)	()	()
5	19	44(0)	0(0)	44(44)	132(0)	220(88)	176(44)	176(132)	()	()
6	19	48(48)	0(0)	0(0)	0(0)	0(0)	48(0)	144(144)	()	()
7	19	52(0)	0(0)	0(0)	52(52)	()	()	()	()	()
8	23	112(112)	56(0)	280(224)	560(224)	1008(280)	()	()	()	()
9	24	60(0)	60(0)	60(0)	180(60)	720(300)	()	()	()	()
10	20	64(64)	0(0)	0(0)	64(64)	64(0)	0(0)	96(96)	256(128)	()
11	27	68(68)	408(0)	()	()	()	()	()	()	()
12	21	72(72)	0(0)	0(0)	0(0)	0(0)	0(0)	216(216)	144(0)	()
13	19	76(76)	0(0)	0(0)	0(0)	0(0)	0(0)	0(0)	76(76)	76(76)
14	25	160(80)	240(80)	240(240)	400(160)	()	()	()	()	()
15	27	84(84)	84(84)	756(168)	518(182)	()	()	()	()	()
16	28	264(264)	88(0)	440(264)	()	()	()	()	()	()
17	23	92(92)	0(0)	0(0)	0(0)	0(0)	276(92)	()	()	()
18	28	96(0)	96(0)	288(0)	288(96)	624(336)	()	()	()	()

Table 13.3 WiMax 2/3*A* LDPC Codes

i	S_{min}	$N_{s_{min}}$	$N_{s_{min}+1}$	$N_{s_{min}+2}$	$N_{s_{min}+3}$	$N_{s_{min}+4}$	$N_{s_{min}+5}$	$N_{s_{min}+6}$
13	15	76(76)	228(152)	()	()	()	()	()
14	14	80(0)	80(80)	160(0)	()	()	()	()
15	15	84(84)	252(0)	()	()	()	()	()
16	15	88(88)	0(0)	()	()	()	()	()
17	15	92(92)	0(0)	92(92)	460(276)	()	()	()
18	15	96(96)	0(0)	96(96)	480(384)	()	()	()

Table 13.4 WiMax 2/3*B* LDPC Codes

i	S_{min}	$N_{s_{min}}$	$N_{s_{min}+1}$	$N_{s_{min}+2}$	$N_{s_{min}+3}$	$N_{s_{min}+4}$	$N_{s_{min}+5}$	$N_{s_{min}+6}$
6	16	96(48)	432(48)	()	()	()	()	()
7	15	52(52)	0(0)	104(104)	156(104)	728(312)	2041(533)	()
8	16	63(63)	56(56)	196(56)	560(168)	1568(196)	()	()
9	17	120(60)	()	()	()	()	()	()
10	15	64(64)	0(0)	0(0)	0(0)	128(0)	384(64)	()
11	18	204(68)	()	()	()	()	()	()
12	15	72(72)	0(0)	0(0)	72(0)	()	()	()
13	15	76(76)	0(0)	0(0)	0(0)	0(0)	76(0)	()
14	16	80(80)	80(0)	()	()	()	()	()
15	15	84(84)	0(0)	0(0)	0(0)	84(84)	294(168)	()
16	16	88(88)	88(0)	()	()	()	()	()
17	20	92(92)	92(0)	92(0)	()	()	()	()
18	15	96(96)	0(0)	0(0)	0(0)	0(0)	144(96)	()

of intersected unknown bits. Let $w(\mathbf{h}_j^{I_j(F)})$ be the weight of ones on jth column of $\mathbf{H}$, which is the number of active rows intersected with jth column. Under a worst case assumption, the $I_j(F)$ with larger column weight of ones on jth column is always with value 1, then the active rows can be compensated by $I_j(F)$ and the total number of active rows ϕ is reduced by $w(\mathbf{h}_j^{I_j(F)})$ until $\phi \leq 0$. Algorithm 13.3 shows the pseudocode of computing the smallest number of intersected unknown bits q in order to produce no active rows. The lower bound $w'(F) = w(F) + q$ is the result.

Algorithm 13.3 Simple method to find the smallest collection set of active rows

1. Arrange the set of $\mathscr{J}(F)$ in descending order, where $\mathbf{h}_{i'_0}$ is the column with the maximal column weight corresponding to constraint F.
2. q is initialised as 0.

while $\phi > 0$ **do**

1). ϕ is subtracted by $w(\mathbf{h}_{i'_0})$.
2). q is accumulated by 1.

end while

Table 13.5 WiMax 3/4*A* LDPC Codes

i	S_{min}	$N_{s_{min}}$	$N_{s_{min}+1}$	$N_{s_{min}+2}$	$N_{s_{min}+3}$	$N_{s_{min}+4}$	$N_{s_{min}+5}$	$N_{s_{min}+6}$
6	10	48(0)	0(0)	24(0)	240(48)	624(288)	()	()
7	12	26(0)	156(52)	260(104)	2184(416)	()	()	()
8	12	28(0)	112(0)	224(168)	952(280)	()	()	()
9	12	90(60)	60(0)	180(60)	372(192)	()	()	()
11	12	34(0)	68(68)	0(0)	0(0)	()	()	()
12	12	36(0)	0(0)	0(0)	0(0)	72(0)	504(144)	()
13	12	38(0)	76(76)	0(0)	76(76)	()	()	()
14	12	40(0)	80(0)	160(0)	240(0)	240(0)	800(160)	()
15	12	42(0)	0(0)	0(0)	0(0)	0(0)	168(84)	()
16	12	44(0)	0(0)	0(0)	88(88)	()	()	()
17	12	46(0)	0(0)	0(0)	0(0)	0(0)	0(0)	()
18	12	48(0)	0(0)	0(0)	0(0)	0(0)	0(0)	96(0)

Table 13.6 WiMax 3/4*B* LDPC Codes

i	S_{min}	$N_{s_{min}}$	$N_{s_{min}+1}$	$N_{s_{min}+2}$	$N_{s_{min}+3}$	$N_{s_{min}+4}$	$N_{s_{min}+5}$	$N_{s_{min}+6}$
7	9	52(52)	52(52)	52(52)	312(156)	988(416)	3094(1274)	11180(3952)
8	12	560(392)	616(224)	1792(616)	7784(2968)	()	()	()
9	10	60(60)	60(60)	130(10)	540(240)	2190(810)	7440(2940)	()
10	11	64(64)	128(128)	128(64)	960(640)	3648(1408)	()	()
11	13	272(204)	748(544)	2992(1564)	()	()	()	()
12	12	72(0)	576(432)	576(216)	2520(936)	()	()	()
13	12	228(228)	380(304)	988(836)	2888(836)	()	()	()
14	10	80(80)	0(0)	0(0)	0(0)	640(480)	2416(1216)	()
15	11	84(0)	84(84)	336(168)	546(294)	1260(588)	()	()
16	14	176(88)	968(792)	()	()	()	()	()
17	13	184(92)	92(92)	1012(644)	()	()	()	()
18	12	16(16)	96(96)	672(480)	()	()	()	()

13.2.2 Best Next Coordinate Position Selection

In the evaluation of the lower bound above, the selected unconstrained positions are assumed to have value 1. Correspondingly, the first position in the index list has maximal column weight and is the best choice for the coordinate to add to the constraint set F.

Table 13.7 Weight Spectra and stopping set spectra for the WiMax LDPC Codes [1]

	Code Length $N = 576 + 96i$									
i	0	1	2	3	4	5	6	7	8	9
N	576	672	768	864	960	1056	1152	1248	1344	1440
Code Rate	Minimum Codeword Weight d_m									
1/2	13	19	20	19	19	21	19	22	23	27
2/3A	10	9	8	11	13	10	14	13	14	13
2/3B	12	11	14	16	15	15	16	15	16	17
3/4A	10	10	10	12	12	13	13	13	14	12
3/4B	8	8	9	11	11	9	11	9	12	10
5/6	5	7	7	7	7	7	7	7	7	7
	Minimum Stopping Set Weight s_m									
1/2	18	18	18	21	19	19	24	19	24	24
2/3A	10	10	11	9	12	13	13	14	14	14
2/3B	10	12	13	15	14	16	16	18	18	17
3/4A	9	8	10	11	12	12	10	12	12	12
3/4B	9	10	10	10	11	11	11	12	12	12
5/6	6	6	7	7	7	7	7	9	7	8
	Code Length $N = 576 + 96i$									
i	10	11	12	13	14	15	16	17	18	
N	1536	1632	1728	1824	1920	2016	2112	2208	2304	
Code Rate	Minimum Codeword Weight d_m									
1/2	20	27	21	19	25	27	28	23	31	
2/3A	12	13	15	15	15	15	15	15	15	
2/3B	15	18	15	15	16	15	16	20	15	
3/4A	14	13	17	13	17	17	15	20	19	
3/4B	11	13	13	12	10	12	14	13	12	
5/6	7	7	8	8	7	7	8	8	9	
	Minimum Stopping Set Weight s_m									
1/2	24	28	28	28	25	29	29	28	28	
2/3A	15	12	14	16	14	16	17	18	18	
2/3B	19	18	18	20	17	20	17	21	20	
3/4A	12	12	12	12	12	12	12	12	12	
3/4B	13	13	12	13	14	11	14	13	15	
5/6	8	9	7	9	7	8	9	8	10	

13.3 Results

The algorithms above have been used to evaluate all of the low-weight stopping sets for some well-known LDPC codes. The results are given in Table 13.1 together with the respective references where details of the codes may be found. The total

number of stopping sets are shown for a given weight with the number of codewords in parentheses. Interestingly, the Tanner code has 93 parity-check equations, 2 more than the 91 parity-check equations needed to encode the code. If only 91 parity-check equations are used by the iterative decoder there is a stopping set of weight 12 instead of 18 which will degrade the performance of the decoder. The corollary of this is that the performance of some LDPC codes may be improved by introducing additional, dependent, parity-check equations by selecting low-weight codewords of the dual code. A subsequent tree search will reveal whether there has been an improvement to the stopping sets as a result.

13.3.1 WiMax LDPC Codes

WiMax LDPC codes [1], as the IEEE 802.16e standard LDPC codes, have been fully analysed and the low-weight stopping sets for all combinations of code rates and lengths have been found. Detailed results for WiMax LDPC codes of code rates $1/2, 2/3A, 2/3B, 3/4A, 3/4B$ are given in Tables 13.2, 13.3, 13.4, 13.5, 13.6. In these tables, the code index i is linked to the code length N by the formula $N = 576 + 96i$. The minimum weight of non-codeword stopping sets (s_m) and codeword stopping sets (d_m) for all WiMax LDPC codes is given in Table 13.7.

13.4 Conclusions

An efficient algorithm has been presented which enables all of the low weight stopping sets to be evaluated for some common LDPC codes. Future research is planned that will explore the determination of efficient algorithms for use with multiple computers operating in parallel in order to evaluate all low weight stopping sets for commonly used LDPC codes several thousand bits long.

13.5 Summary

It has been shown that the indicative performance of an LDPC code may be determined from exhaustive analysis of the low-weight spectral terms of the code's stopping sets which by definition includes the low-weight codewords. In a breakthrough, Rosnes and Ytrehus demonstrated the feasibility of exhaustive, low-weight stopping set analysis of codes whose parity-check matrix is sparse using a bounded tree search over the length of the code, with no distinction between information and parity bits. For an (n, k) code, the potential total search space is of size 2^n but a good choice of bound dramatically reduces this search space to a practical size. Indeed, the choice of bound is critical to the success of the algorithm. It has been shown that an improved

algorithm can be obtained if the bounded tree search is applied to a set of k information bits since the potential total search space is initially reduced to size 2^k. Since such a restriction will only find codewords and not all stopping sets, a class of bits is defined as unsolved parity bits, and these are also searched as appended bits in order to find all low-weight stopping sets. Weight spectrum results have been presented for commonly used WiMax LDPC codes in addition to some other well-known LDPC codes.

An interesting area of future research has been identified whose aim is to improve the performance of the iterative decoder, for a given LDPC code, by determining low-weight codewords of the dual code and using these as additional parity-check equations. The tree search may be used to determine improvements to the code's stopping sets as a result.

References

1. WiMax LDPC codes, Air interface for fixed and mobile broadband wireless access systems, IEEE Std 802.16e-2005 (2005). http://standards.ieee.org/getieee802/download/802.16e-2005.pdf
2. Wang, C.C., Kulkami. S.R., Poor, H.V.: Exhausting error-prone patterns in LDPC codes (2007). http://arxiv.org/abs/cs.IT/0609046, submitted to IEEE Transaction in Information Theory
3. Hu, X.Y., Eleftheriou, E., Arnold, D.M.: Regular and irregular progressive edge-growth tanner graphs. IEEE Trans. Inf. Theory **51**(1), 386–398 (2005)
4. Lan, L., Zeng, L., Tai, Y.Y., Chen, L., Lin, S., Abdel-Ghaffar, K.: Construction of quasi-cyclic LDPC codes for AWGN and binary erasure channels: a finite field approach. IEEE Trans. Inf. Theory **53**, 2429–2458 (2007)
5. MacKay, D.: Encyclopedia of sparse graph codes [online] (2011). http://www.inference.phy.cam.ac.uk/mackay/codes/data.html
6. Tanner, R.M., Sridhara, D., Fuja, T.: A class of group-structured LDPC codes. In: Proceedings of the International Symposium on Communications Theory and Applications (ISCTA) (2001)
7. Rosnes, E., Ytrehus, O.: An algorithm to find all small-size stopping sets of low-density parity-check matrices. In: International Symposium on Information Theory, pp. 2936–2940 (2007)